THESIS

MICROGRID CONTROL STRATEGY UTLIZING THERMAL ENERGY STORAGE WITH RENEWABLE SOLAR AND WIND POWER GENERATION

by

Kevin L.J. Hawxhurst

June 2016

Thesis Advisor:	Anthony Gannon
Co-Advisor:	Andrea Holmes

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE MICROGRID CONTROL STRATEGY UTLIZING THERMAL ENERGY STORAGE WITH RENEWABLE SOLAR AND WIND POWER GENERATION	**5. FUNDING NUMBERS**
6. AUTHOR(S) Kevin L.J. Hawxhurst	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Project supported by the Office of Naval Research's (ONR) Energy Systems Technical Evaluation Program (ESTEP) supported by Dr. Richard Carlin and under the technical monitoring of Stacey Curtis.	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

As part of the Department of Defense's exploration into alternative energy solutions, this research focused on developing and implementing a control strategy for a microgrid system that was developed using a multi-physics energy approach. The objective was to demonstrate a microgrid system that more effectively uses renewable energy based on the end-use application of energy. The NPS Integrated Multi-Physics Renewable Energy Laboratory microgrid system was designed primarily for heating and cooling applications and utilizes thermal storage capabilities.

A novel control strategy was also implemented to decrease the need for backup electrical power. The control strategy matches load demand from a chiller and heater to power generation from renewable solar and wind resources. Energy is stored as ice for cooling applications and in high temperature ceramic bricks for heating applications.

A controller was designed using MATLAB and successfully implemented the desired control strategy. This was challenging as communication between the controller, the microgrid, the loads, and the thermal storage devices had to be established across multiple architectures. Using MATLAB, the controller operated nearly continuously for six months, collecting data for analysis. This research proves that the end-use energy design concept works by putting in place a working demonstration plant.

14. SUBJECT TERMS microgrid, control strategy, renewable energy, thermal storage, multi-physics, end-use energy	**15. NUMBER OF PAGES** 125
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

MICROGRID CONTROL STRATEGY UTLIZING THERMAL ENERGY STORAGE WITH RENEWABLE SOLAR AND WIND POWER GENERATION

Kevin L.J. Hawxhurst
Ensign, United States Navy
B.S., United States Naval Academy, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2016

Approved by: Anthony Gannon
Thesis Advisor

Andrea Holmes
Co-Advisor

Garth Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As part of the Department of Defense's exploration into alternative energy solutions, this research focused on developing and implementing a control strategy for a microgrid system that was developed using a multi-physics energy approach. The objective was to demonstrate a microgrid system that more effectively uses renewable energy based on the end-use application of energy. The NPS Integrated Multi-Physics Renewable Energy Laboratory microgrid system was designed primarily for heating and cooling applications and utilizes thermal storage capabilities.

A novel control strategy was also implemented to decrease the need for backup electrical power. The control strategy matches load demand from a chiller and heater to power generation from renewable solar and wind resources. Energy is stored as ice for cooling applications and in high temperature ceramic bricks for heating applications.

A controller was designed using MATLAB and successfully implemented the desired control strategy. This was challenging as communication between the controller, the microgrid, the loads, and the thermal storage devices had to be established across multiple architectures. Using MATLAB, the controller operated nearly continuously for six months, collecting data for analysis. This research proves that the end-use energy design concept works by putting in place a working demonstration plant.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
COM	Communications
COTS	Commercial off the Shelf
DOD	Department of Defense
ESTEP	Energy Systems Technology Evaluation Program
FOB	Forward Operation Base
FY	Fiscal Year
ID	Identifier
IMPREL	Integrated Multi-Physics Renewable Energy Laboratory
NI cDAQ	National Instruments Compact Data Acquisition
NPS	Naval Postgraduate School
ONR	Office of Naval Research
SOC	State of Charge
POC	Percent of Completion
PV	Photovoltaic
TCP/IP	Transmission Control Protocol/Internet Protocol
UINT32	Unsigned 32 Bit Integer
USB	Universal Serial Bus
VAWT	Vertical Axis Wind Turbine
VFD	Variable Frequency Drive
VRLA	Value Regulated Lead Acid
VSC	Variable Speed Compressor
WBI	Wind Box Inverter

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge my advisor, Anthony Gannon, and my co-advisor, Andrea Holmes, for all of their help throughout my research. Ms. Holmes spent countless hours helping me better understand the equipment that was used for this project. I was fortunate to have such supportive advisors that made for a very rewarding research experience. I would also like to thank my brother, Christopher Hawxhurst, for all of his technical revisions of my thesis, my girlfriend, Ashley Eves, for editing my thesis, and Tim Bihl for lending his expertise in communication protocols.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This project falls under the Energy Systems Technology Evaluation Program (ESTEP). The program provides funding by the Office of Naval Research (ONR) for energy-related projects at the Naval Postgraduate School (NPS). It was created in FY 2013 in response to the Secretary of the Navy Ray Mabus's call for greater use of renewable energies [1]. The U.S. Department of Defense (DOD), the nation's largest consumer of energy, "has launched several initiatives to reduce fossil fuel use by improving energy efficiency and shifting to renewable energies since 2010" [2]. This research is part of a larger effort by the DOD to meet increasing energy demands with secure and sustainable energy technologies.

A. MOTIVATION

The energy industry has largely remained the same since Thomas Edison commissioned the first commercial power grid in lower Manhattan in 1882 [3]. Although the electrical grid effectively provides energy for a large portion of the population, there are many risks and disadvantages associated with the current grid. According to the Department of Energy (DOE), demand for electricity has outpaced transmission rates by 25% every year since 1982 [4]. As demand for energy increases, new energy solutions are going to have to be found outside of the traditional electric grid. Buildings that need greater energy security, developing counties, and facilities in remote locations are examples of markets that require unique energy solutions. As new technologies increases, more energy solutions become available. The next step is to design and implement more effective energy systems using available technology.

B. TRADITIONAL ENERGY APROACH

The electric grid is a network of generating stations that produce electric energy, high-voltage transmission lines that transport energy, and distribution lines that provide energy to customers [5]. The traditional electric grid provides energy to customers by matching power generation to demand. Power generation and transmission capabilities must be much greater than the average demand in order to provide sufficient power during periods of peak demand. Most existing generating stations deliver electricity to customers at an overall fuel-to-electricity efficiently in the range of 28–32%; this represents a loss of about 70% of the primary energy provided to the generator [6]. Renewable energy resources struggle to fit the current model due to intermittency, despite their potential to generate large amounts of power [7]. The lack of any large-scale energy storage capabilities means that there is no buffer between power generation and demand. The traditional electric grid is becoming an inefficient and inflexible method of providing energy.

C. MULTI-PHYSICS ENERGY APPROACH

The multi-physics energy approach was developed at NPS by Gannon and Pollman in 2015 [7]. The approach is a methodology of designing the generation, transport, and storage of energy systems using all available technology based on the end-use application of energy [7]. It also suggests a novel concept of using demand side management to match load demand to power generation, rather than the opposite, more traditional supply side approach. The multi-physics approach was used to design the Integrated Multi-Physics Renewable Energy Laboratory (IMPREL). IMPREL is an experiment energy system that was designed to demonstrate different methods of generating, storing, and transporting energy from renewable resources [7]. It was commissioned in 2015 at NPS in Monterey, CA.

D. OBJECTIVE

The objective of this research was to demonstrate an energy system designed using the multi-physics approach that could more effectively use renewable energy resources based on the end-use application of energy. This required developing and implementing a control strategy for the IMPREL microgrid that could match load demand to power generation from renewable energy resources. The goal of this project was to provide a unique energy system that can be used by the DOD and other communities, while validating the multi-physics and end-use energy approach.

E. CHALLENGES

The practical implementation of a microgrid controller required coordination across multiple communications protocols and the practical constraints of many systems. In particular, the batteries were very vulnerable to damage from cycling. If load demand drastically exceeded power generation for any significant period, the batteries would completely discharge and be permanently damaged. When developing a controller that adjusts power to loads and relies on multiple communications all working together, there were many possibilities for scenarios to go dangerously wrong. Therefore, protecting the equipment from potential damage was always the top priority. Developing the controller itself was also a challenge. The controller was required to implement an untraditional control strategy that not been attempted or demonstrated anywhere within the DOD or by current microgrid research efforts.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The IMPREL energy system is an experimental microgrid that aims to use renewable energy resources more effectively by utilizing targeted energy storage. This project is part of a larger effort by the DOD to research and implement new energy technologies.

A. DEPARTMENT OF DEFENSE ENERGY OBJECTIVES

By some accounts, the U.S. Department of Defense (DOD) is the largest consumer of petroleum in the world, consuming 117 million barrels of oil in FY 2011, and accounting for 71% of the DOD energy use in FY2010 [8]. Despite decreasing petroleum use 4% from FY2005 to FY2011, spending on petroleum increased from $4.5 billion to $17.3 billion during the same period due to rising costs [8]. Concerns over rising costs, the risks associated with the military's dependence on petroleum, and the need for more secure and sustainable energy has led to both federal legislation and executive orders directing the DOD to reduce its use of petroleum. This has directed an extensive alternative energy effort by the DOD to improve energy efficiency, enhance energy security, and cut installation and operational energy costs. DOD renewable energy projects have already increased 43% from 2010 to 2012 and are anticipated to increase exponentially over the next 20 years [9].

According to the DOD, about 75% of DOD's energy is used for operational purposes and about 25% is used in installations [10]. There are several federal initiatives that have contributed to the decline in the DOD's installation energy usage. The National Defense Authorization Act of 2010 requires that DOD facilities use 25% renewable energy by 2025 [2]. In its FY 2013 *Operational Energy Annual Report*, the DOD also outlined several initiatives to reduce operational energy use. The DOD has already invested in deploying and generating energy at the location of military operations to reduce the need to transport fuel long distances [11].

Forward operation bases (FOB) are one example of where the U.S. military is trying to deploy and generate energy on-site. These bases often rely on flexible microgrid solutions to provide energy. However, generators are often used at sub-optimal loads due to the variability of military operations. The goal of the United States Marine Corps' (USMC) Expeditionary Energy Concepts (E2C) program, created by the USMC Commandant in 2009, is to quickly evaluate and deploy technologies that reduce the need for fuel supply-train. One of the technology areas that E2C is focusing on for 2016 is "energy storage technology for mobile electric microgrid application" [12]. Sandia National Labs (SNL) is also working with the United States Marines Corps by developing the microgrid design tool (MDT). The MDT will help determine the optimal mix of energy technologies to deploy to FOBs to meet the specific requirements of that operation [13].

B. MICROGRIDS

A microgrid is a localized electric grid that can operate autonomously. Microgrid systems usually include a combinations of power generations sources, loads, and storage devices that can all be centrally controlled. Figure 1 illustrates an example of a microgrid system. The microgrid manager is the centralized controller that integrates and optimizes power generation with the controllable loads and allows the system to operate autonomously [14]. Therefore, these systems can be disconnect from the centralized electric grid (macrogrid) and can operate in an isolated mode, usually referred to as *islanding*.

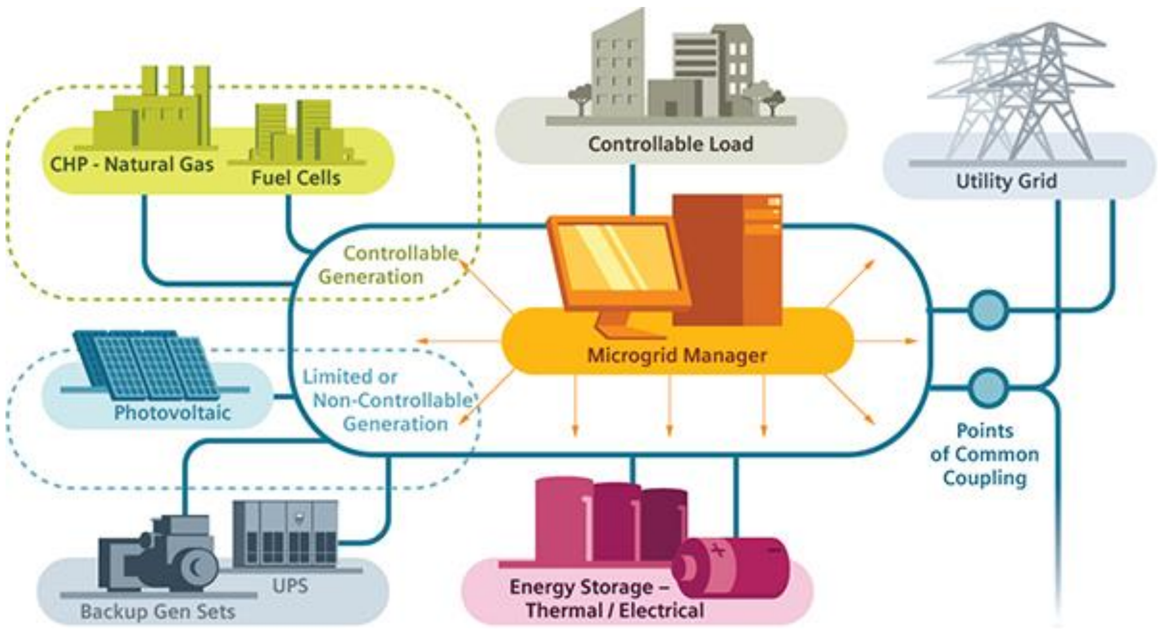


Figure 1. Microgrid Illustration. Source: [14]

As described by Lasseter [6], microgrids operate using a peer-to-peer and plug-and-play model for each component of the microgrid. In the peer-to-peer model, no microgrid components are critical for operation. This allows generators or loads to separate from the system while maintaining service. The plug-and-play model implies that a generator or load can be added to the system without changes to the controls. This flexibility is provided by the inverters, which automatically convert between DC and AC power. The microgrid manager uses information from all the components to provide system optimization.

The flexibility of microgrids means that they are often used with renewable energy resources. In addition to their independence and reliability, this allows microgrids to provide unique energy solutions. However, microgrid systems still use the traditional energy approach, matching power generation to load demand. The need to provide power during peak demand, coupled with the intermittent nature of renewable energy resources, means that current microgrid system are still dependent on electrical storage and backup generation.

C. RENEWABLE ENERGY

Renewable energy resources, according to the Department of Energy (DOE), include solar, biomass, wind, geothermal, and water [15]. These resources are naturally regenerative. Photovoltaic (PV) solar cells and wind turbines are two examples of the many technologies that generate power using renewable energy. The DOD is interested in utilizing renewable energy for military operations due to its potential for energy security and independence, both strategic and operational. Renewable energy has the ability to produce on site power, which would improve the energy security and sustainability of operations. However, renewable energy resources tend to be intermittent and unpredictable by nature. Therefore, power generation from these resources is not always available. The power density of renewable energy is also significantly lower than petroleum, which means that more equipment is need to provide the same amount of power.

D. THERMAL ENERGY STORAGE

Energy storage devices provide a buffer to an electric grid. The storage devices can absorb energy from the generators when power generation exceeds load demand and feed that energy back onto the grid when load demand exceeds power generation. Energy storage devices are often used with renewable energy resources to compensate for the intermittent power generation. The most common energy storage devices are batteries, which store electrical energy. Thermal storage devices, which stores thermal energy as heat, are less common.

Water is commonly used for thermal storage due to its high heat capacity, which allows it to store large amounts of heat. Water can be used for both hot and cold thermal storage. In cold storage, water is frozen to form ice. Ice storage takes advantage of water's latent heat of fusion (334 kJ/kg), which is the amount of heat required to change phases from a liquid to a solid [16]. However, water cannot be heated past 100°C (212°F) at atmospheric pressure without creating steam, making it difficult to use for high temperature heat storage. Ceramic bricks are another material used for thermal storage. They are used for hot thermal storage due to their ability to store heat at very high temperatures, in excess of 1400°F [17]. Figure 2 shows the energy densities of ice and ceramic bricks compared to common battery technologies.

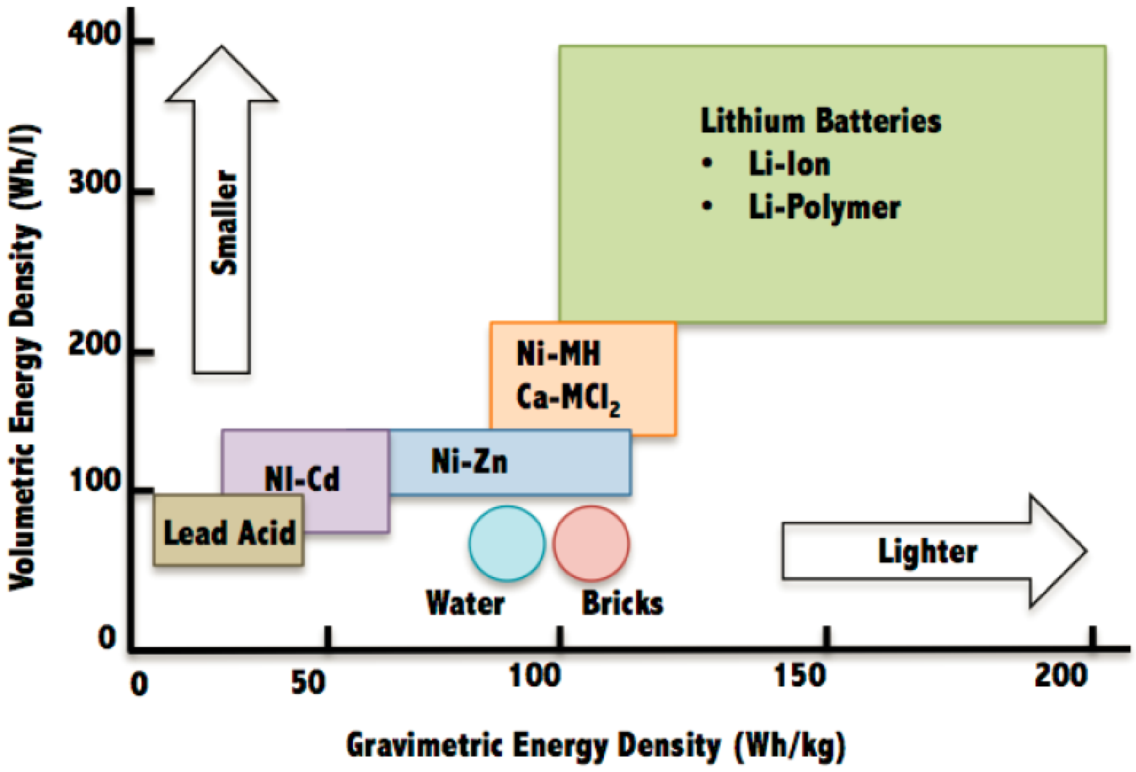


Figure 2. Energy Densities of Different Materials. Source: [18].

In addition to having equivalent energy densities, thermal storage has several advantages over batteries. Thermal storage can directly provide heating and cooling, without converting thermal energy back to electrical energy. The devices have much greater lifetimes since they can be cycled several thousand times with minimal degradation. Batteries have limited lifetimes, and their energy storage capacity decreases over time. Thermal storage devices are also simpler than batteries, and therefore tend to be much cheaper. Comparing the installation cost per kWh of thermal storage to batteries, ceramic bricks storage is 21% cheaper than the least expensive battery technology [16]. Ice storage is 73% cheaper than the least expensive battery technology [18]. Table 1 summarizes the cost and cycle life comparisons between thermal storage and batteries.

Table 1. Comparison of Energy Storage. Adapted from [16], [18], [19].

Storage Method	**Thermal Storage**		**Battery Storage**		
	Ice	**Ceramic Bricks**	**Alkaline-cell**	**Lead-acid**	**Lithium-ion**
Cost per kWh	$37	$110	$140	$150	$400
Cycle Life	10+ yrs	10+ yrs	20	300	1000

According the DOE, the majority of energy consumed in commercial and residential buildings goes towards heating or cooling. Figure 3 shows the consumption of energy in commercial and residential buildings. The figure breaks down the total energy consumption into its end-uses, such as space heating, water heating, space cooling, and refrigeration, which are all forms of thermal energy.

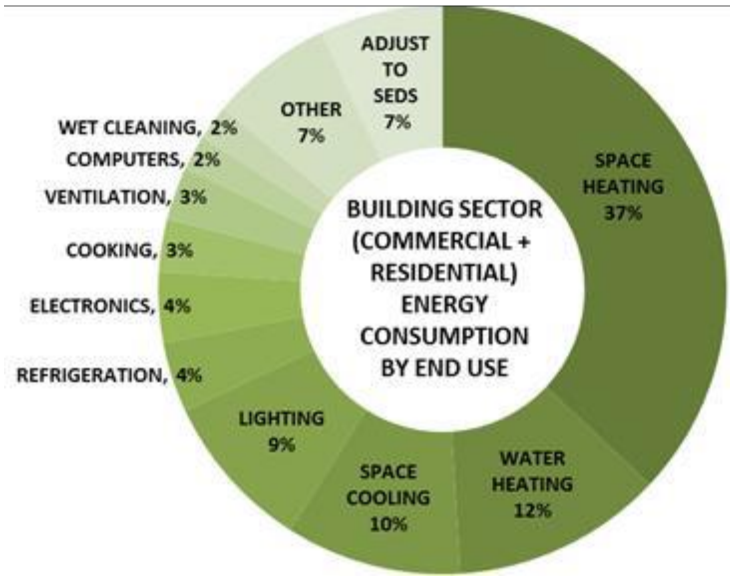


Figure 3. Energy Consumption by End Use. Source: [20].

E. INTEGRATED MULTI-PHYSICS RENEWABLE ENERGY LABORATORY

The Integrated Multi-Physics Renewable Energy Laboratory (IMPREL) is an experimental microgrid system that was developed at NPS using the multi-physics approach. The multi-physics concept, which proposes different methods of generating, transporting, and storing energy based on the end-use application of energy, is illustrated in Figure 4. The figure illustrates the traditional approach alongside the multi-physics approach of providing energy to electrical, heating, and cooling loads. The end-use energy would therefore be electrical loads as well as heating and cooling.

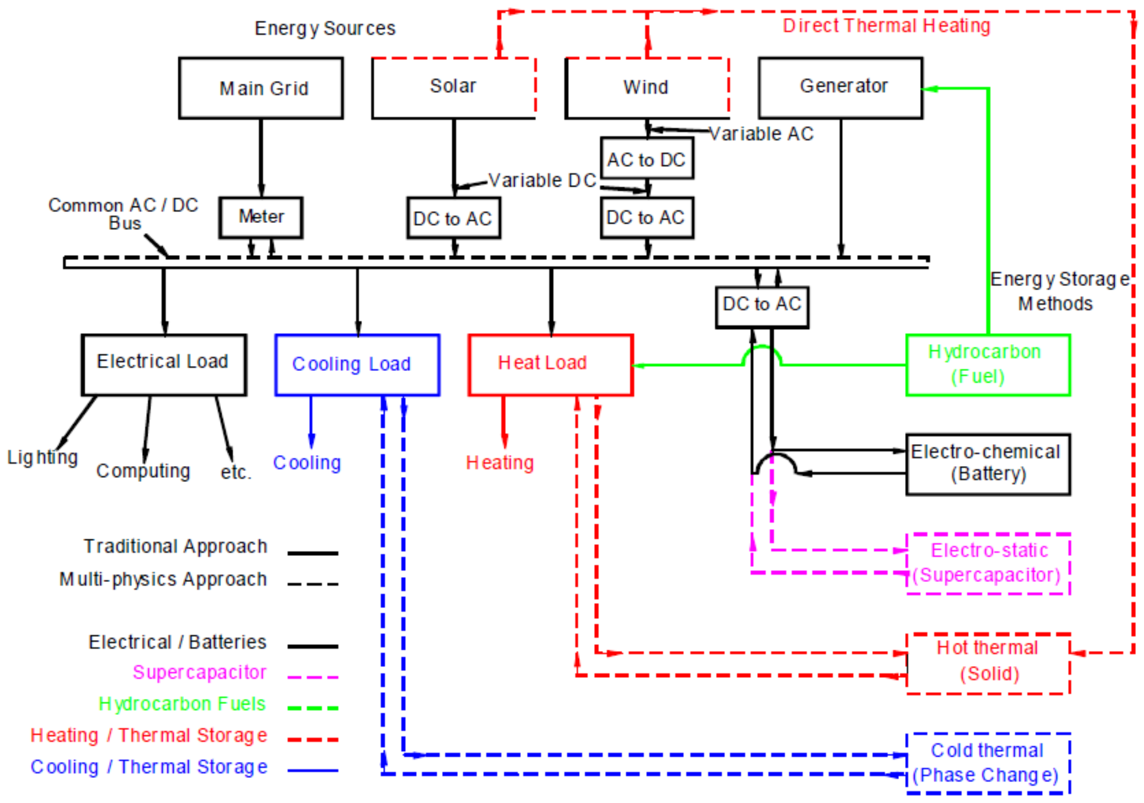


Figure 4. Multi-Physics Theory. Source: [7].

The solid lines represent the traditional approach to energy, whereas the dotted lines represent alternative methods. The traditional approach uses the electric grid and hydrocarbons as energy sources. Energy is transported using electricity to electrical, heating, and cooling loads. The multi-physics approach proposes using renewable solar and wind resources for power generation. Energy is transported using both electricity and direct thermal heating. Electrical energy covers electrical and cooling loads while direct thermal heating covers heating loads. Excess energy generated by the renewable resources is stored as electricity in supercapacitors or batteries or as hot or cold thermal energy in thermal storage devices. The thermal storage devices can be used directly to augment heating and cooling demands, without converting energy back to electricity.

The IMPREL microgrid system was designed primarily for heating and cooling applications based on the end-use energy approach. The system requires energy storage capabilities since it relies solely on renewable energy generation. Therefore, a combination of thermal storage devices and battery storage was used. Ice was used for cold thermal storage and ceramic bricks were used for hot thermal storage. Since the majority of end-use energy is thermal energy, the thermal storage capacity was designed to make up the majority of the systems energy storage capabilities.

The IMPREL microgrid was attempts to increase overall efficiency of an energy systems by utilizing power generation from renewable solar and wind resources more effectively. The proposed control strategy matches load demand to power generation, storing excess energy in electrical and thermal storage devices. The IMPREL concept is shown in Figure 5. Renewable wind and solar energy provides power generation, and the microgrid transports electrical energy. Batteries store the excess electrical energy from the microgrid, and a heater system and chiller system store hot and cold thermal energy, respectively [7].

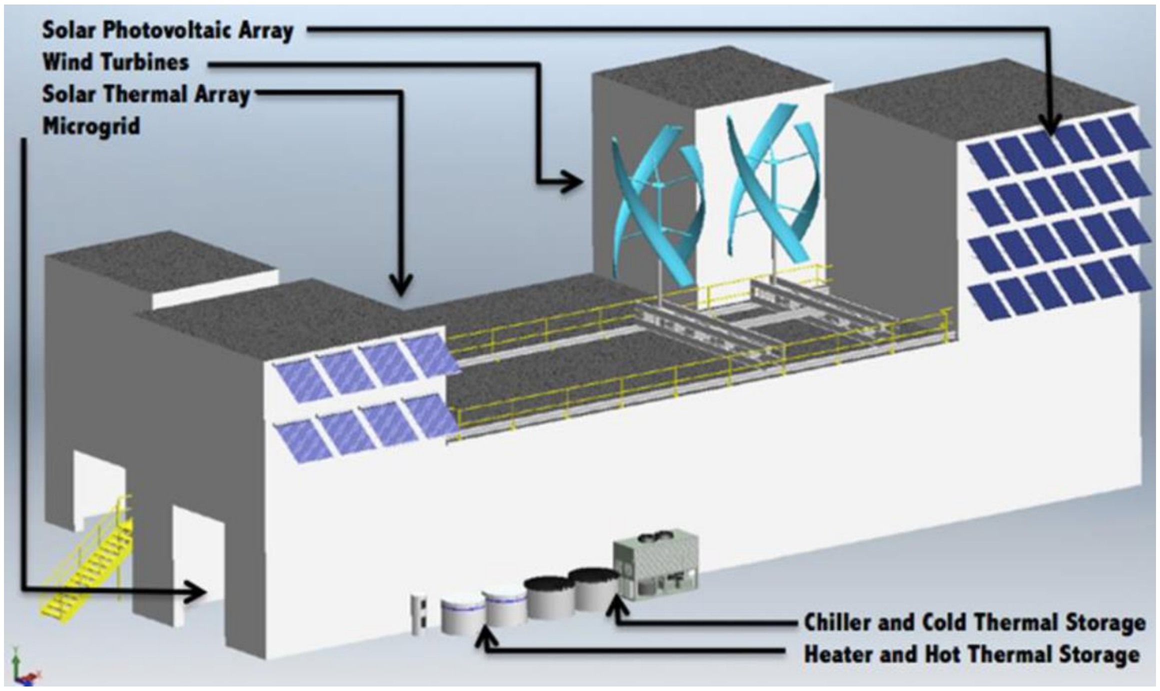


Figure 5. IMPREL Concept. Source: [7].

To implement the proposed control strategy that integrated different components of the IMPREL microgrid system, a custom controller is designed and developed. The controller is required to operate autonomously, prioritize loads, match load demand to power generation, and provide heating and cooling to building spaces. The controller will allow the microgrid system to respond to heating and cooling demands, as well as store excess energy in storage devices.

THIS PAGE INTENTIONALLY LEFT BLANK

III. EXPERIMENTAL METHODS AND EQUIPMENT

This section describes the components and equipment of the IMPREL microgrid system, the communications network for the controller, and the desired control strategy.

A. EQUIPMENT

All of the components used for the microgrid system are commercial off the shelf (COTS) equipment. The following three sections cover the equipment used for power generation, transport, and energy storage. Use of COTS equipment simplifies design of the system and ensures easy reproduction. Overriding the default controls of the heating and cooling units was the only customization of the equipment. This was necessary to fully implement the control strategy. Otherwise, default controls would interfere with adjustments made by custom controller since equipment operates differently than intended.

1. Power Generation

The microgrid uses purely renewable resources for power generation. Power is generated from two VisionAIR 3.2kW vertical axis wind turbines (VAWT) and 6.4kW of photovoltaic (PV) solar panels, shown in Figure 6. The microgrid also has the option to draw power from the main grid. However, this capability was not used for this research, completely isolating the microgrid from the main grid.

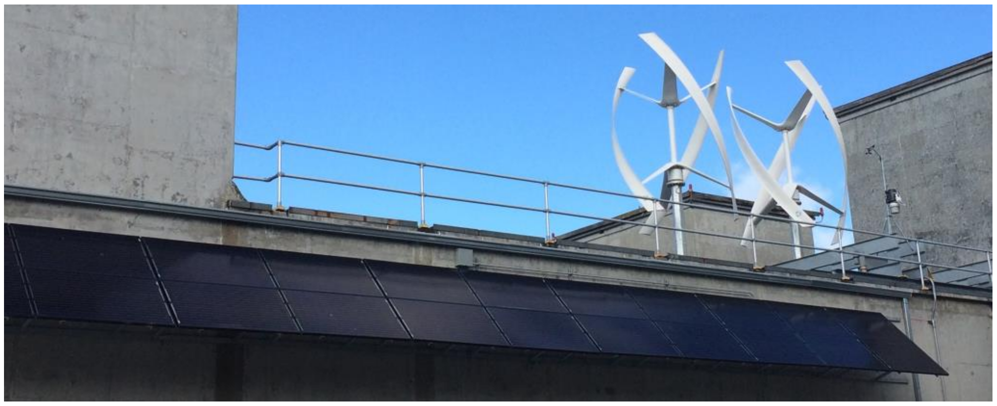


Figure 6. Vertical Axis Wind Turbines and Photovoltaic Solar Panels

a. Solar Energy

The 6.4kW of solar power is produced by twenty-four PV solar panels, each of which provide 265W of power. The panels are dived between two groups of twelve panels each, designated group A and group B. This allows at least one group to provide power if some of the panels become shaded. Each group provides DC power to a single SMA Sunny Boy DC to AC inverter, shown in Figure 7. The inverter connects to the three-phase microgrid.

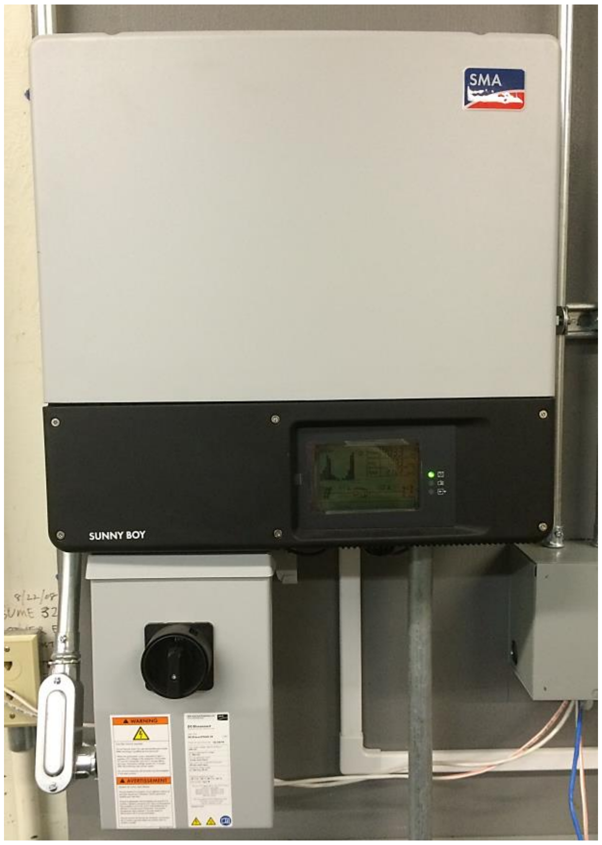


Figure 7. Solar Sunny Boy Inverter

b. Wind Energy

Two wind turbines output wild AC power, which varies in frequency and voltage, to individual inverters, shown in Figure 8. Aurora Wind Box Inverters (WBI) convert the wild AC power to variable DC power. Two SMA Sunny Boy inverters convert the variable DC power from the Aurora boxes back to AC power. If there is gusting or high wind speeds, excess power is diverted to a resistor bank. The resistor banks dissipate excess energy as heat and act as a large load on the generators, reducing the speed of the wind turbines.

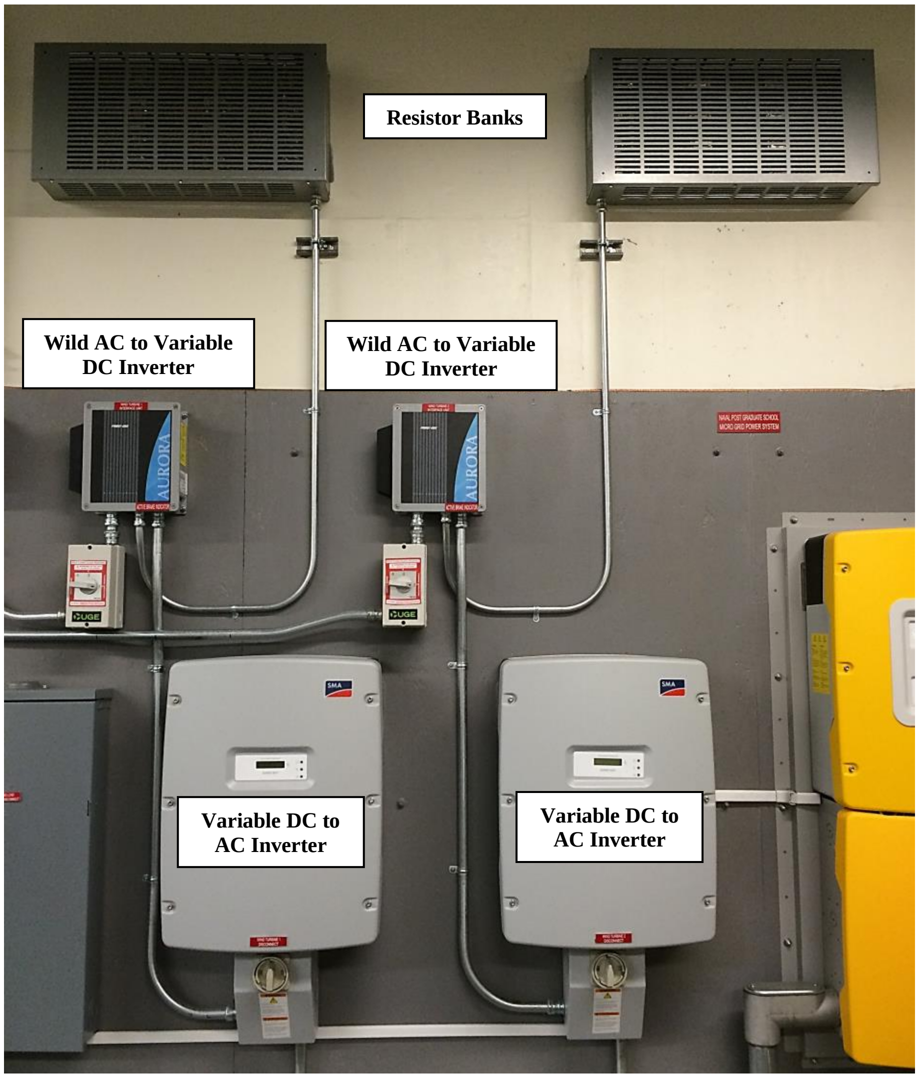

Resistor Banks
Wild AC to Variable DC Inverter
Wild AC to Variable DC Inverter
Variable DC to AC Inverter
Variable DC to AC Inverter

Figure 8. Wind Turbine Inverters

2. Energy Transport

An electric microgrid consisting of three SMA Sunny Island inverters creates a three-phase system at 208V, shown in Figure 9. They are connected as one master inverters and two slave inverters. The three-phase microgrid synchronizes each power generation source to the microgrid, and then transports the energy to the leads.

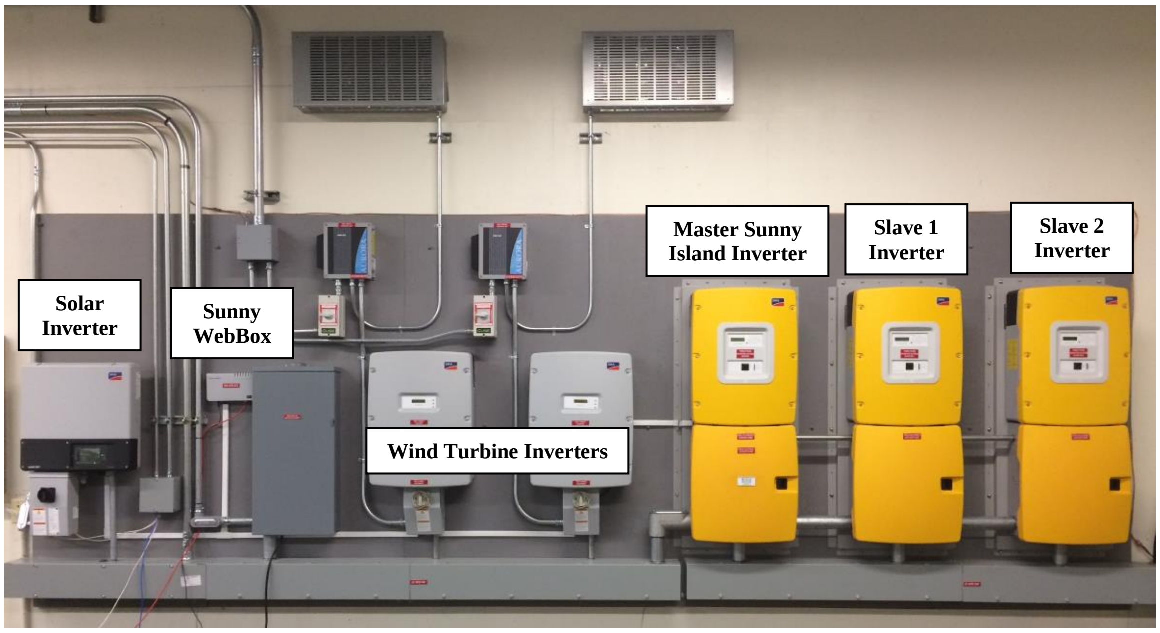


Figure 9. SMA Three-Phase Microgrid

The microgrid consists of six inverters: one solar Sunny Boy inverter, two wind turbine Sunny Boy inverters, and three Sunny Island inverters. The inverters communicate with a central SMA Sunny WebBox, shown in Figure 10, which generates a webpage with all the data and status information from the microgrid. The WebBox is connected to a computer with a TCP/IP interface. This allows the user to remotely monitor the system. However, remote connection to the microgird from outside IMPREL has been disabled.

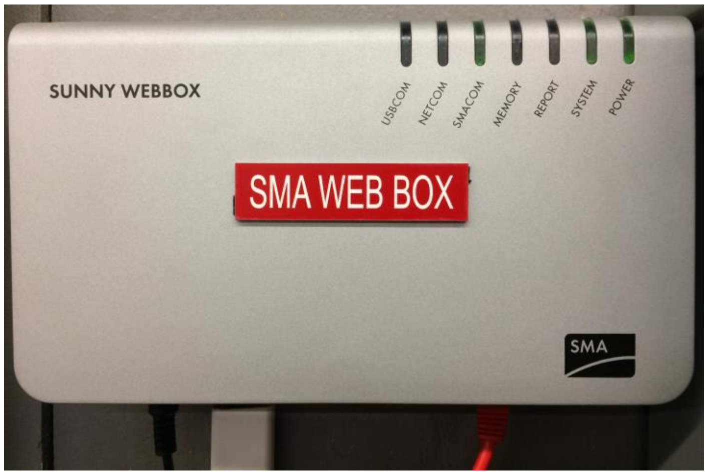


Figure 10. SMA Sunny WebBox

3. Energy Storage

Energy storage for the microgrid system is divided between electrical storage and thermal storage. Electrical energy storage provides backup power to the microgrid system when the renewable resources are not generating power. The electrical energy stored in a battery bank also feeds power onto the grid when electric load demand exceeds the power generation. Thermal storage from a heater and chiller system is used directly for heating and cooling applications respectively. The thermal storage systems convert electrical energy into thermal energy and store energy until it is needed.

a. Battery Bank

The battery bank, shown in Figure 11, stores electrical energy for the microgrid. The battery bank is composed of twenty-four valve-regulated lead-acid (VRLA) gel cells with a total storage capacity of 800–1000Ah, equivalent to approximately 50kWh. The batteries are used to stabilize the grid but are not intended to run the loads directly for extended periods. The electrical capacity of the batteries is relatively small, and running the loads from batteries would drain them within a few hours.

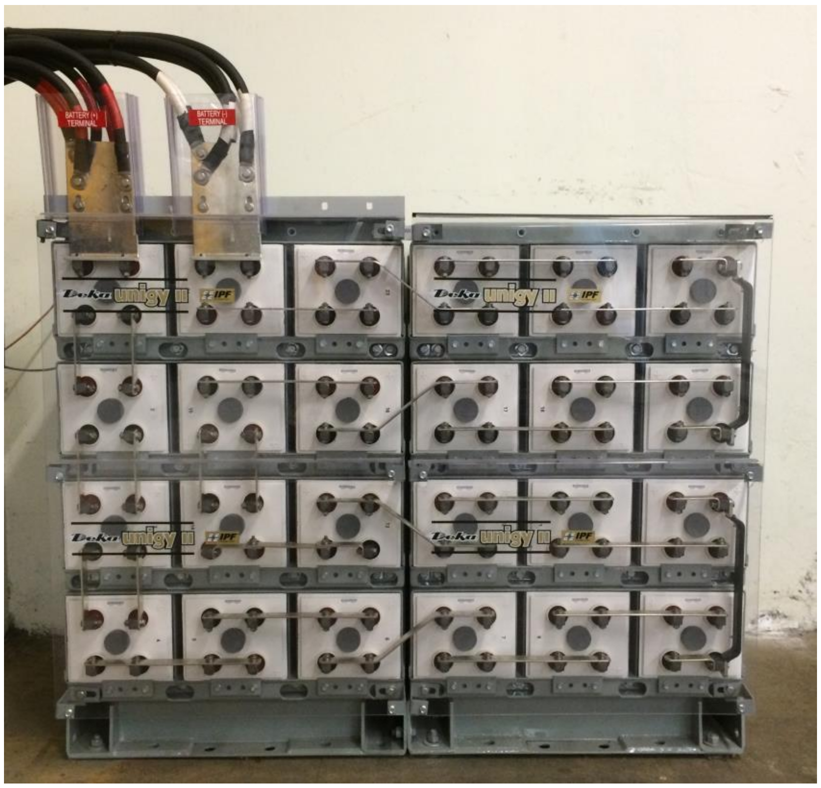


Figure 11. VRLA Battery Bank

b. Cold Thermal Storage

The cold thermal storage system, show in Figure 12, uses a 7½-ton Trane chiller with a variable speed compressor (VSC) and variable frequency drive (VFD) pump. The chiller cools water containing 25% propylene glycol to -4°C (25°F) and circulates it through heat exchangers in a CALMAC IceBank storage tank, shown in Figure 13. The water-glycol solution freezes the water surrounding the heat exchanger inside the tank. This process extracts the heat from the water in the IceBank until approximately 95% of the water inside the tank has been converted to ice [21]. The IceBank can provide 170kWh of energy storage (48.5ton-hours of cooling). When there is a demand for cooling, air would be vented through the IceBank, melting the ice and cooling the air. This requires the installation of a liquid air heat exchanger between the glycol mixture and air, which has not been installed on this system.

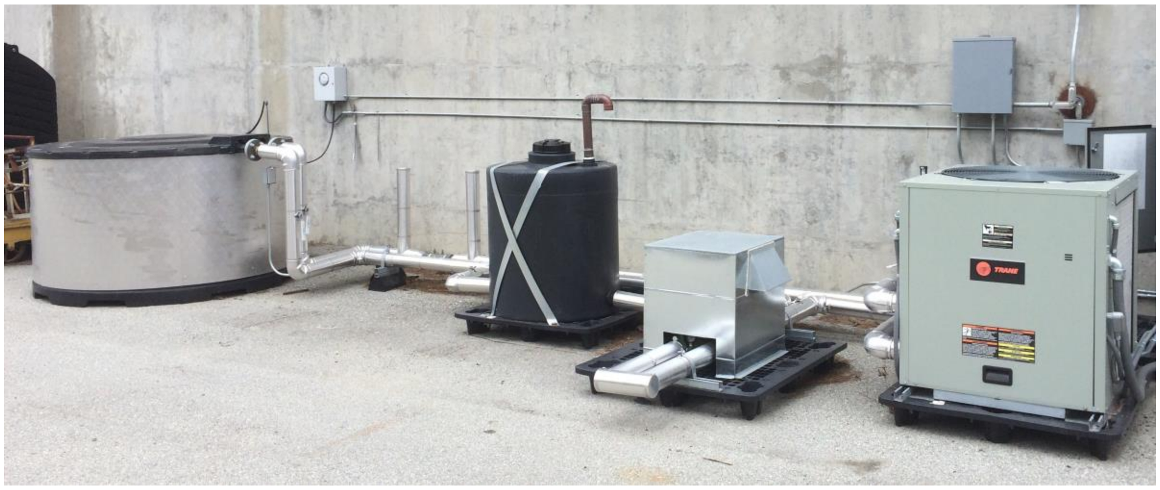


Figure 12. Trane Chiller and Cold Thermal Storage System

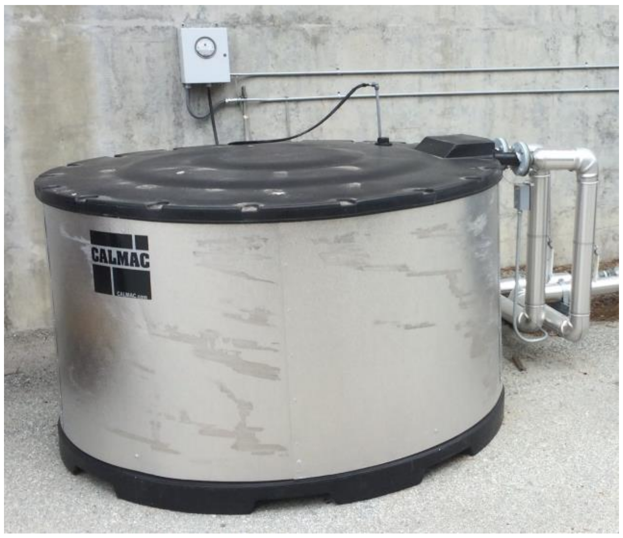


Figure 13. CALMAC IceBank

According to CALMAC, ice-making de-rates the nominal chiller capacity by approximately 30–35% [21]. The system was designed to run at night, when electricity costs are lower and the lower temperatures would keep the unit operating efficiently. The control strategy for this project prioritizes the chiller for direct cooling to the building spaces, which is done at the normal thermodynamic efficiency. Once the cooling load has been answered, the chiller is used to generate ice for thermal storage, which does cause the chiller to operate in a less efficient mode. However, the chiller uses excess power generation from the renewable resources to generate ice, so there is no cost associated with the decreased efficiency.

c. Hot Thermal Storage

The hot thermal storage system used a Steffes Comfort Plus forced air heating system, shown in Figure 14. The 4200 series heater was designed to store off-peak electricity in the form of heat, the same design concept of the cold thermal storage system. The heater converts electricity into heat within an insulated ceramic brick core, Figure 15. Since electricity is converted to heat within the insulated core, the energy conversion is theoretically 100% efficient. The heater can reach a maximum temperature of 650°C (1200°F), providing 120kWh of energy storage (409,440BTU of heating). When there is a demand for heat, a fan blows air through the ceramic bricks, which heats the air. Locating this system inside or under the space that is to be heated ensures residual heat loss is not wasted.

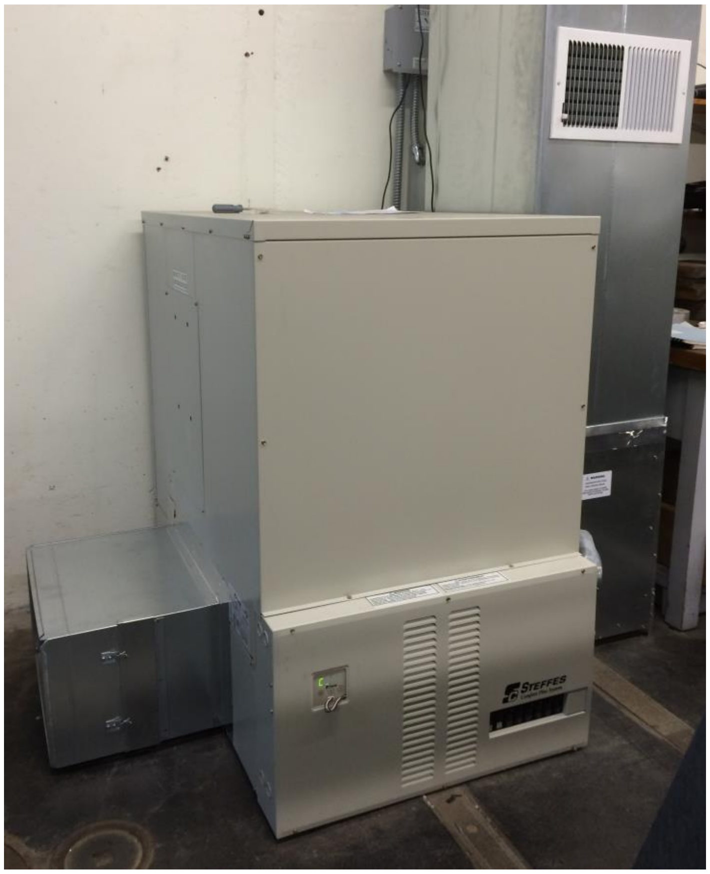


Figure 14. Steffes Heater

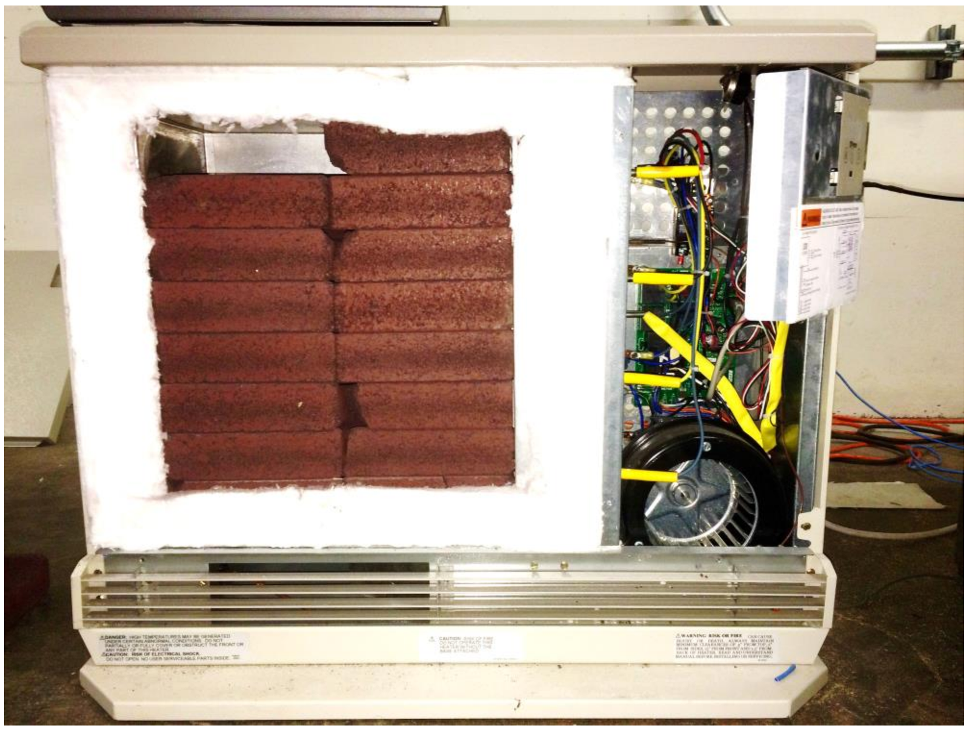


Figure 15. Ceramic Bricks used for Hot Thermal Storage

B. COMMUNICATIONS

The installation of the equipment for the microgrid system was completed during previous NPS thesis research projects [16], [18], [22]. The components, however, had not been integrated into a working system. In order to implement the desired control strategy, communications had to be established between a centralized controller and the equipment. The initial focus of this research was therefore setting up a communications network to be used by the centralized controller. Figure 16 illustrates this communications network.

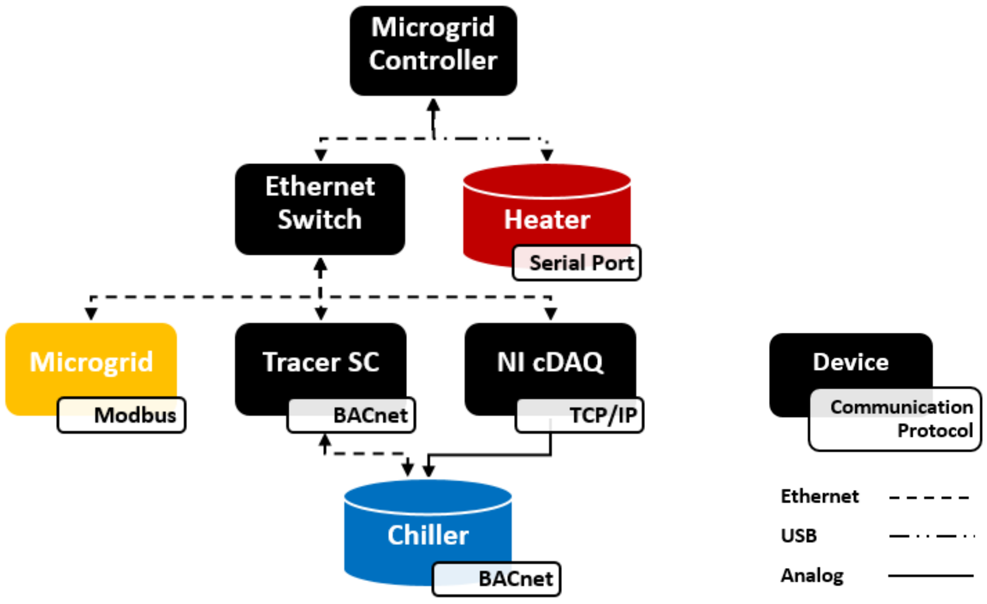
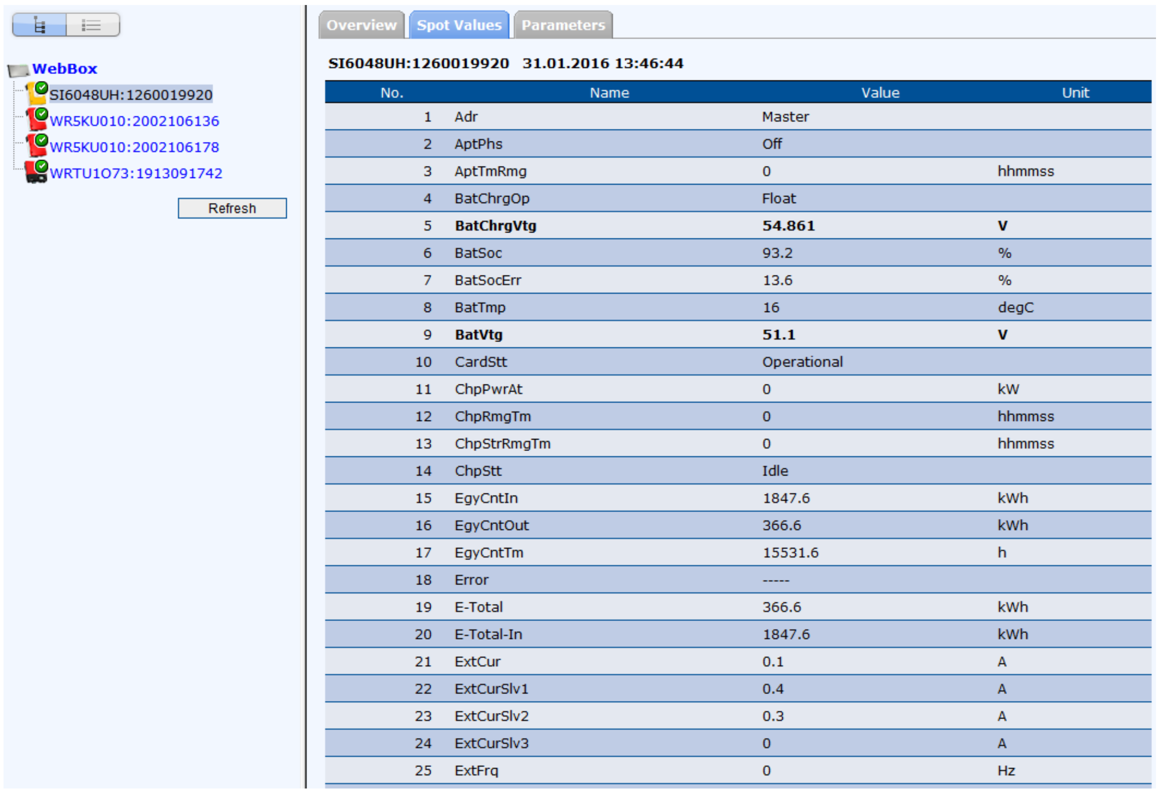


Figure 16. Communications Network

The central microgrid controller uses a USB serial port adapter to communicate with the heater system, Modbus communications protocol over Ethernet to communicate with the microgrid inverters, and BACnet communications protocol over Ethernet to communicate with the chiller system. However, since BACnet commands are disabled, an analog output device is used to vary the chiller power. MATLAB was used to develop functions that could execute each of these communications protocols. A separate function was developed for each protocol.

1. SMA Sunny WebBox Webpage

The SMA Sunny WebBox collects information from the inverters and uses that data to generate a webpage. Information on the webpage updates every 30 seconds. The WebBox connects to the computer via Ethernet. Figure 17 shows an example of data collected from the master Sunny Island inverter that is displayed by the webpage.



SI6048UH:1260019920 31.01.2016 13:46:44

No.	Name	Value	Unit
1	Adr	Master	
2	AptPhs	Off	
3	AptTmRmg	0	hhmmss
4	BatChrgOp	Float	
5	**BatChrgVtg**	**54.861**	**V**
6	BatSoc	93.2	%
7	BatSocErr	13.6	%
8	BatTmp	16	degC
9	**BatVtg**	**51.1**	**V**
10	CardStt	Operational	
11	ChpPwrAt	0	kW
12	ChpRmgTm	0	hhmmss
13	ChpStrRmgTm	0	hhmmss
14	ChpStt	Idle	
15	EgyCntIn	1847.6	kWh
16	EgyCntOut	366.6	kWh
17	EgyCntTm	15531.6	h
18	Error	-----	
19	E-Total	366.6	kWh
20	E-Total-In	1847.6	kWh
21	ExtCur	0.1	A
22	ExtCurSlv1	0.4	A
23	ExtCurSlv2	0.3	A
24	ExtCurSlv3	0	A
25	ExtFrq	0	Hz

Figure 17. Example Sunny Island Data on Webpage

Data is read from the web page using a URL filter MATLAB script that scans the page for given variables. The URL filter function has numerous drawbacks. It takes over two minutes to scan the necessary data, sometimes the filter scans the wrong values, and if the user navigates the webpage while the function is scanning it will cause an error. The webpage also has to be logged in for the function to work. The function was required to scan the page a minimum of every five minutes to remain logged in to the website. The URL Filer function was eventually replaced with the Modbus function as it avoids many of these issues.

2. Modbus Protocol

The SMA microgrid uses Modbus communications protocols. The following information from the SMA Sunny WebBox Modbus Interface describes the variation of the Modbus communications protocol for SMA products.

> **Modbus Protocol**
>
> The Modbus Application Protocol is an industrial communication protocol that is currently mainly used in the solar sector for plant communication in PV power stations.
>
> The Modbus protocol has been developed for reading data from or writing data to clearly defined data areas. The Modbus specification does not specify what data is within which data area. This information must be defined specifically for a device in a so-called Modbus profile. With knowledge of the specific Modbus profile, a Modbus master (e.g., a SCADA system) can access the data of a Modbus slave (e.g. Sunny WebBox).
>
> The SMA Modbus profile is the special Modbus profile for SMA devices.
>
> **SMA Modbus Profile**
>
> The SMA Modbus profile contains definitions for selected SMA devices. For the definition there was a reduction of the available data and an assignment to the respective Modbus registers. The SMA Modbus profile contains for example overall and daily energy, current output, voltages and currents. The assignment between SMA device data and Modbus addresses is divided into sections in the SMA Modbus profile that can be addressed by Unit IDs.
>
> In order to enable access to data of an SMA device, a special gateway is required that is provided by Sunny WebBox.
>
> **Plant Topology**
>
> The SMA Modbus profile has been designed for a hierarchical plant structure. This structure contains the WebBox as communication device that is equipped with a Modbus TCP/IP interface. All other SMA devices that are connected to the WebBox via the SMA fieldbus are subordinate to it.
>
> From the perspective of the Modbus protocol, the WebBox is a Modbus slave that provides a gateway to SMA devices. The SMA devices can only be addressed using this gateway per Unit ID. [23]

Modbus communications was incorporated into the control scrip in order to improve the communications between the controller and the microgrid. It allows the controller to bypass the webpage generated by the Sunny WebBox and request data directly from the Sunny Boy and Sunny Island inverters. This dramatically speeds up communications, allowing access to data as it becomes available. For example, the webpage displays data for all of the inverters, updating approximately every 30 seconds. When polling the inverters directly, however, some values update as frequently as every 5 seconds.

Using Modbus communications, the controller polls data from the three Sunny Boy inverters and the master Sunny Island inverter. Each inverter has a specified Unit Identifier (ID). Within each inverter, data is stored in specified register addresses. Most values are stored in two adjacent registers as unsigned 32-bit integers (UNIT32). The controller converts these values to decimal values. The query and response of all the data takes less than two seconds. The controller only polls the inverters every 20 seconds, allowing time for all the values to update. Incorporating Modbus communications has drastically improved the performance of the controller.

3. Serial Port

The Steffes heater uses a universal serial bus (USB) serial port adapter for communications. The MATLAB control scrip communicated with the heater using a USB connection. Using a communications (COM) port, requests could be sent to the heater to adjust the element on percent, fan setting, and read the temperature as well as the element on percent and fan setting.

A serial port is a general-purpose serial communications interface. Serial ports use the RS-232 telecommunications standard for data transmission. Most modern computers no longer have serial ports and require a serial-to-USB converter to interface with RS-232 serial devices [24]. A virtual COM port (VCP) driver from FTDI Chip was used to convert the USB device to an additional COM port. The USB device could then be accessed as a standard COM port using application software such as MATLAB [25].

4. BACnet Protocol

The Trane chiller system uses BACnet communications protocol. The Building Automation and Control Networks (BACnet) is an American national standard data communications protocol developed by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) [26]. The following information on BACnet protocol is from the Trane Integration Guide for BACnet Communication Interfaces.

> The Building Automation and Control Network (BACnet and ANSI/ASHRAE Standard 135–2004) protocol is a standard that allows building automation systems or components from different manufacturers to share information and control functions. BACnet provides building owners the capability to connect various types of building control systems or subsystems together for a variety of reasons. In addition, multiple vendors can use this protocol to share information for monitoring and supervisory control between systems and devices in a multi-vendor interconnected system.
>
> The BACnet protocol identifies standard objects (data points) called BACnet objects. Each object has a defined list of properties that provide information about that object. BACnet also defines a number of standard application services that are used to access data and manipulate these objects and provides a client/server communication between devices. [27]

The Trane chiller system uses a Tracer UC600 Programmable controller along with two unit controllers, a UC600 and a UC400. The following information on these controllers is from the Trane website.

> The Tracer® SC is an intelligent field panel that communicates with unit controllers (BACnet) that provide standalone control of HVAC equipment. The Tracer® SC scans all unit controllers to update information and coordinate building control, including building subsystems such as VAV and chilled water systems. The LAN allows building operators to manage these varied components as one system using web access. [28]
>
> Trane offers programmable BACnet controllers that can be customized to specific needs. The Tracer UC400 and Tracer UC600, which are also used as airside controllers, are compatible with a wide variety of application scenarios, and support graphical programming and the BACnet protocol. These two programmable BACnet controllers are designed to work with Tracer SC and third-party BACnet MS/TP systems. [29]

The Tracer SC controller is similar to the SMA Sunny Webbox in that it receives and stores data from the unit controllers. The Tracer SC controller uses this data to generate a webpage and to update the Trane control panel. It monitors data from the entire system, including compressor and pump speeds, glycol-water temperatures, and ice storage levels The Tracer SC controller is also intended to implement controls, but that feature was not used for this project.

BACnet communications was integrated into the controller to allow the controller to poll data from the Tracer SC controller. The controller can also communicate directly with the two unit controllers. The UC400 controller controls the compressor and pump speeds. The controller can send BACnet commands to the UC400 controller to control these speeds. However, turning off the control feature of the Tracer SC override this feature as well. Therefore, speeds commands are sent to the UC400 controller using an analog signal.

5. NATIONAL INSTRUMENTS COMPACT DATA ACQUISITION

A National Instruments (NI) compact data acquisition (cDAQ) device is a modular platform that can be used to interface with different sensors and signals. The NI cDAQ is used with the NI 9263 analog output module to send analog signals to the chiller. These signal are received by the UC400 controller and modulate compressor speed. This adjusts the power to the chiller. The NI cDAQ chassis and analog output module are shown in Figure 18. The module is used to output 0–10V, which corresponds to 0–100% compressor speed.

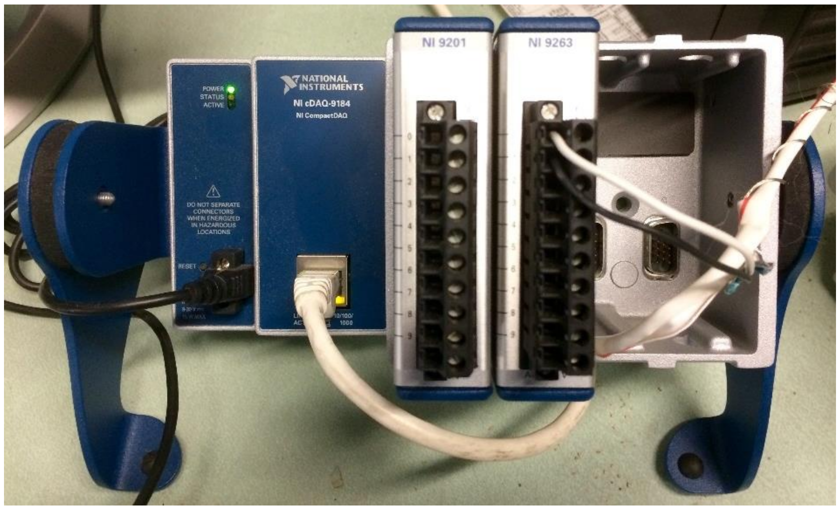


Figure 18. NI cDAQ Analog Output Device

C. CONTROL STRATEGY

The controller was designed and implemented using MATLAB, a matrix-based numerical computing program that offers extensive capabilities. The microgrid controller is composed of multiple elements, shown in Figure 19. Each element was designed and run using MATLAB. The microgrid controller shutdown and microgrid load controller are control loops that run simultaneously and continuously. The communications functions allow the load controller to communicate with the microgrid system.

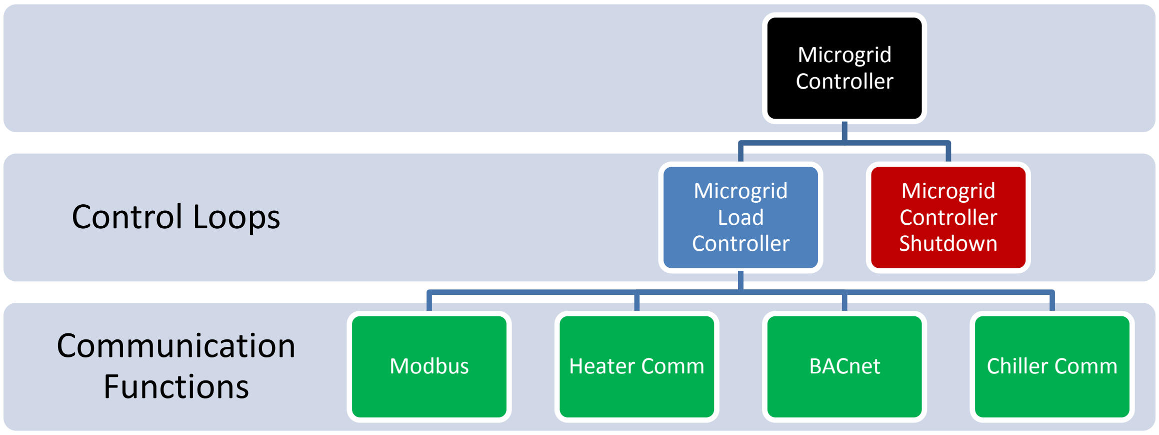


Figure 19. Microgrid Controller Hierarchy

1. Microgrid Controller Shutdown

The microgrid controller shutdown provides a safety backup to prevent damage to the batteries and equipment. Ending the load controller loop abruptly would leave the loads running at the power levels last received by the controller. This is obviously an issue, as the loads will eventually drain the batteries, causing damage to the system. The controller shutdown monitors the load controller. If the load controller encounters an issue, such as lost communications, the control loop will terminate and the controller will stop operating. If the microgrid controller shutdown detects that the load controller has stopped operating, it will turn of all of the loads. Although the controller is not infallible, it provides an extra layer or redundancy and safety to the system.

2. Communications Functions

The microgrid load controller utilizes four separate functions to communicate with the microgrid system. Each function executes a different communications protocol, as described in the previous sections, in order to communicate with different equipment. The Modbus function uses Modbus protocols to communicate with the microgrid inverters. The Heater Comm function uses a serial port to communicate and send commands to the heater. The BACnet function uses BACnet protocols to communicate with the chiller system. The Chiller Comm function uses the NI cDAQ to send analog speed commands to the chiller.

3. Microgrid Load Controller

The microgrid load controller implements the desired control strategy using the control loop shown in Figure 20. The objective of the control strategy is to match the load demand from the chiller and heater systems to power generation from the renewable resources, without requiring additional power from the batteries. Control provided by the default SMA software handles power distribution of the microgrid devices. The microgrid load controller varies demand by adjusting load power. By matching load demand to power generation, the system uses all available power from the renewable energy resources. Excess thermal energy, not immediately needed for heating or cooling applications, is stored in the thermal storage devices. Excess electrical energy is stored in the batteries.

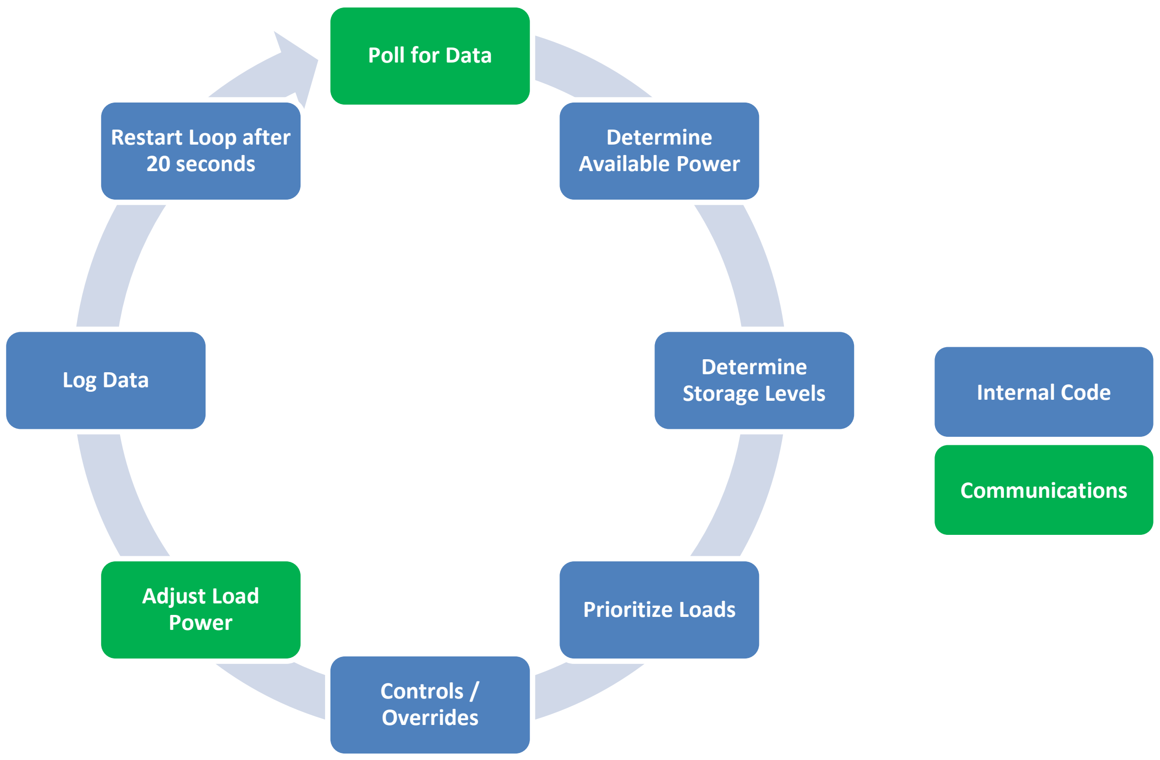


Figure 20. Microgrid Load Controller—Control Loop

For each iteration of the control loop, the Microgrid Load Controller polls data from the microgrid and thermal storage devices, determines available power and thermal storage levels, prioritizes the loads, and adjusts the power to the loads accordingly. Data for each iteration is logged in a text file. There are multiple controls and overrides within the control loop that regulate operations. These controls take of the form of "if/then" MATLAB statement.

The controller constantly adjusts load demand in response to changes in power generation. Since the renewable resources used for power generation are intermittent by nature, the controller cannot predict how much power will be available. The performance of the controller is limited to how quickly it can respond and make adjustments. The control loop is currently limited to 20 seconds, the time it takes the inverters to update their values.

The aforementioned explanation for the control loop of the Microgrid Load Controller (Figure 19) is highly simplified. The following three sections describe the control loop in detail. The MATLAB code for the both control loops and four functions is located in Appendix A.

a. Pre Control Loop

Before the microgrid load controller enters the control loop, it executes preliminary code. This code defines certain variables and sets up the control loop to run more smoothly. This only occurs when the microgrid load controller is first started. The first section of preliminary code allows the user to adjust the controller. The user can choose the start and stop time that the controller will run loads and the minimum power requirements for certain operations. The user also sets the desired thermal storage levels, which the controller uses to determine which load to prioritize. These inputs are further explained in the following section.

The next section of code establishes communications with the equipment. A TCP/IP connection required for Modbus communications is established with the SMA Sunny WebBox. This prevents the Modbus function from establishing a new connection each time it polls a register, which would eventually crash the Sunny WebBox due to the large amount of network traffic. All BACnet devices on the network are also discovered and stored in an address cache. This improves the communications speed of the BACnet function.

In the last section, a text file is created and memory is pre-allocated. All data from the control loop for that day is recorded in the text file. Memory is pre-allocated to improve speed of the control loop. Since the data files within the controller increase in size after every iteration, this prevents MATLAB from allocating larger blocks of memory after each iteration.

b. Control Loop

(1) Time

There are multiple timed elements of the control loop, so the first step is to establish the time. To ensure that each loop takes exactly 20 seconds, a timer is started to time the duration of each loop iteration. The time of day is also established. All variables for that iteration are associated with that time. The hour and minute are also specified for controls that operate depending on the time of day. The date is also established and used to create a new text data file each day.

(2) Modbus Communications

The SMA microgrid communicates using Modbus protocols. MATLAB establishes communications with the microgrid using the Modbus function, which polls data from the inverters via the WebBox using the pre-established TCP/IP connection. The controller specifies the unit ID of the inverter and the register address of the value it wants to poll. UINT32 values are converted to decimal values and the data is extracted to MATLAB.

(3) Calculated Variables

The controller calculates battery power, which is the product of the battery current and battery voltage, and the total power generation, which is defined as the sum of the power generation from the solar panels and the wind turbines. Following references to power generation refer to total power generation. Many of the following controls use these variables.

(4) URL Filter Communications

The URL-filter function can mine data variables from the Sunny WebBox webpage. This process is very slow, requiring a few seconds per variable, and can be inaccurate, mining the wrong value. More importantly, the function will output an error if the webpage logs out due to inactivity or a user navigates the webpage, causing the controller to terminate. After the ability to read values using Modbus communications was incorporated, the URL-filter was no longer required to run the controller.

(5) BACnet Communications

The Trane chiller systems communicates using BACnet protocols. MATLAB communicates with the chiller system using the BACnet function, which can poll data from the Tracer SC, UC600, or UC400 controller. The controller specifies the device ID of the controller and the object, instance, and property of the value it wants to poll. The BACnet function can execute different BACnet commands, such as writing values to the compressor and pump speeds. However, the controllers were setup not to accept BACnet write commands in order to override the default Trane controls. If the controllers were updated to accept BACnet write commands, it would eliminate the need for the NI cDAQ. The NI cDAQ currently regulates compressor speed using an analog signal.

(6) Thermal Storage Levels

Ice storage levels are recorded by the Tracer SC controller, which can be read directly using the BACnet function. Heater storage levels are calculated using Equation 1, with ambient room temperature 20°C (70°F) correlating to 0% and the maximum heater temperature of 650°C (1200°F) as 100%. All values in Equation 1 are in Fahrenheit.

$$\text{Heater Storage} = \frac{\text{Heater Temperature} - 70}{1200 - 70} \times 100 \qquad (1)$$

(7) Power Moving Averages

Moving averages compensate for the intermittence of the renewable resources, which can cause the power generation to change rapidly. Moving averages of the power generation, as well as the battery power, are calculated from the last six iterations. This corresponds to a two minute moving average. These averages are used in multiple controls, and can easily be updated to increase or decrease the duration of the moving average.

(8) Max Power Point Tracking (Override 1)

The maximum power point tracking (MPPT) control is the first of three override controls. If any of the overrides are enabled, the loads are shut off. The specified conditions of each override are outline in this section and the following two sections. The loads will not turn back on until the overrides are disabled.

The MPPT control tracks increasing power generation in the morning. The batteries require a large amount of electrical power to charge every morning, as charge is lost throughout the night. This causes the solar panels to output their maximum power generation. The MPPT control records when power generation has peaked and begins to fall off, indicating that the power delivered to the batteries has leveled off and excess power is available. Once this occurs, the override is disabled for the rest of the day and the loads are allowed to run. This override is turned on by default when the control loop restarts at midnight.

(9) Low Power Shutdown and Standby (Override 2)

The low power control is the second override. It includes two controls, the lower power shutdown and low power standby controls. The low power shutdown control will shut down all of the loads if the average power generation falls below a specified limit. This prevents the controller from running loads at very low renewable power. The loads will not turn back on until the average power generation increases above this limit. The low power standby control will shut down all of the loads for a certain period of time, if the average power generation falls below a certain limit but is still greater that the shutdown limit.

The low power standby control is mainly needed if the only load is the chiller, due to its large startup power. Since the microgrid inverters will not output their max power if there is not enough load on the system, the controller cannot determine the maximum available power without running loads. However, the large startup power of the chiller means that it cannot be run at low power. The low power standby control allows the loads to run for five minutes in order to determine the available power generation. If there is insufficient power, the control will shut the loads down for a set amount of time, before allowing them to run again for five minutes. Ideally, this would only occur a few times before power generation increased enough to run the loads without drawing power from the batteries, or power generation decreased below the shutdown limit and the loads remained shut down.

(10) Battery SOC Warning and Boost Charge (Override 3)

The third override is a combination of two battery controls. The first battery control is a low battery state-of-charge (SOC) warning. If the battery SOC drops below

85%, the control will shut off all of the loads and display a warning. The loads will not turn back on until the battery SOC increases to 86%. This allows a buffer if the SOC drops when the loads are reintroduced.

The second battery control is for boost charge mode. The batteries go into a boost charge mode every few days or if the SOC is low. In this mode, the batteries require more power than in float mode, which is the normal operating mode. This allows them to charge quickly. If heater or the chiller are loaded to the mircogrid during this time the boost charge will not complete. Therefore, the control prevents any loads from running while the battery is in boost charge. Both battery controls help to prevent damage to the batteries and improve their lifetime.

(11) Percent of Completion (POC) and Prioritize Loads

The controller determines which load to prioritize based on the percent of completion (POC) of thermal storage. The POC is determined by the current storage levels and the desired levels set by the user, as shown in Equation 2.

$$\text{Thermal Storage POC} = \frac{\text{Current Storage Level}}{\text{Desired Storage Level}} \times 100 \tag{2}$$

The controller prioritizes the load with the lower POC. However, if the average power generation is below 3kW, the minimum power to run the chiller, the controller will default to the heater regardless of POC. If the desired storage level is zero, the POC is set to 100%. In this case the corresponding load will not run.

(12) Max Load Power Control

The controller limits the maximum power of the loads based on the respective thermal storage POC. The default max power is 100% for the chiller and 50% for the heater, since there is not enough power generation to run the heater above 50% during normal operations. Once the thermal storage POC increases above 80%, the controller decreases the maximum power for that load to prevent overshoot and allow power to be used for other loads. Once the thermal storage POC has reached 100%, the maximum power is set to zero, preventing the load from running.

(13) Battery Power Control

The battery power control determines whether power is available to run the loads. Negative battery power (batteries are charging) indicates there could be available power to run the loads. Positive battery power (batteries are feeding power onto the microgrid) indicates load demand has exceeded power generation. The controller will increase or decrease the loads proportionally in an attempt to stabilize the power to the loads at a point where the renewable resources are providing maximum power to the microgrid but also maintaining a trickle charge on the batteries.

Since there is no way to directly determine how much power is currently available from the sources, the control uses a perturbation analysis to determine the maximum power point. If there is available power, the control incrementally increases the power to the loads. The renewables sources respond by increasing power output until they reach their maximum power output. If load demands exceeds power generation, the control responds by decreasing power to the loads. Load power stabilizes when the batteries are being trickle charged at 200–400W. This control relies on the fact that the batteries will always want to accept more than 400W. If power generation increases while the loads are stable, charge to the batteries will exceed 400W as they attempt to draw more power. This signals that there is renewable power available, and that power is then diverted to the loads.

(14) Low Power Startup

The low power startup control assists turning on the loads if no loads are currently running. Since the inverters do not output power when there are no loads, the control mistakenly thinks there is low power. Therefore, it will not run any loads. This control is similar in behavior to the low-power standby control, except it utilizes the heater, which can run at very low power, and there is no time limit. The control sets heater power to one percent if there is at least 150W going to the battery. Normally, the controller would not increase load power unless there was an excess of 400W to the battery. If there is available power, the controller will continue increasing load power. Otherwise, the controller will shut off load power.

(15) High Battery SOC Control

The high battery SOC control decreases the amount of power to the batteries as the SOC increases, since the batteries draw less power. Below 94% SOC, the battery power is set to 200–400W. Once SOC equals 94%, the controller only allows 100–200W of power to the batteries. At 98% SOC, the controller only allows 0–100W.

(16) Low Battery Power Control

The low battery power control decreases the load power if the average battery power is too low. This occurs if the load power is continually too high, which may be caused by multiple reasons. For example, as power generation decreases in the evening, if the control does not decrease load demand quick enough, the average power to the batteries will drop. If the batteries are at a high SOC, this control does not go into effect.

(17) Decrease Power to Non-Prioritized Load

This control only takes affect if the chiller and heater are running simultaneously. If both loads are running, and the controller needs to decrease load power, it will decrease power to the non-prioritized load. This control causes the non-prioritized load to become the prioritized load, since the controller only adjust power to the prioritized load.

(18) Increase Power to Non-Prioritized Load

This control takes affect when the prioritized load is running at maximum power. If this occurs, and there is still available power, the controller will increase power to the non-prioritized load. This control causes the non-prioritized load to become the prioritized load, since the controller only adjust power to the prioritized load.

(19) Adjust Prioritized Load

This control adjust the power of the prioritized load based on the previous controls. The non-prioritized load is not adjusted, and will continue running at the same power.

(20) Overshoot Control

The overshoot control prevents the controller from increasing or decreasing load power two iteration in a row. The inverter values update every 20 seconds, which is the duration of one loop. However, those values do not reflect changes in load power until 40 to 60 seconds, which corresponds to two or three iterations of the control loop. Increasing load power each iteration or decreasing load power each iteration causes load demand to drastically over or under shoot power generation. This created an undesirable affect where the load power would oscillate around the available power. To prevent this, the overshoot control would only allow the controller to make changes to load power every other iteration, or every 40 seconds instead of every 20 seconds. This allows enough time for the data to reflect changes in load power.

(21) Max and Min Power Control

The maximum and minimum power control prevents load power from exceeding its set limits. For the heater, the minimum power is zero and the default maximum power is 50% for the large heater and 100% for the small heater. Since the chiller cannot run at low power, its minimum power is either zero or 22%, depending on the adjustments made to load power. The default maximum power is 100%. If the thermal storage POC increases, or the desired storage is set to zero, the maximum power is lowered or set to zero, respectively.

(22) Overnight Standby

The overnight standby turns of the loads for the night, based on the start and stop times specified by the user. The low power control would eventually turn off the loads for the night. With this control, the loads are usually turned off a few hours before sunset to allow the batteries to finish charging. It also prevents the loads from turning on at night if the wind turbines generate power.

(23) Load Communications

In this section, the controller uses the communications functions to send the determined power commands to the loads. Due to the large startup power of the chiller, if the chiller is being turned on the heater power is set to zero. The chiller is run at lower power and the heater is left off for the first nine iterations, or three minutes. This allows the chiller to turn on and the power to stabilize.

(24) Print and Log Data

This section displays certain data for the screen for reference. All of the data for that iteration is also recorded in the text file for that day.

(25) Restart Control Loop

If the date is the same as the last iteration, the control loop continues to the next iteration. However, if the date has changed, this section restarts the control loop. The text file for the previous day is closed and a new text file is created. The pre-allocated memory for all the variables is then reset. The overrides and loop counter are also reset. The control loops then continues to run.

(26) Pause Loop

This section ends the timer started at the beginning of the loop. This time corresponds to the duration of the loop. The controller is paused for the difference of 20 seconds and the duration of the loop. This ensures that the loops run exactly every 20 seconds.

(27) End Control Loop

The final section offers an optional condition to end the control loop and terminate the controller. The only condition set to end the control loop is if the control loop duration exceeds 40 seconds. This occurs if communications with equipment are not operating properly or equipment is unresponsive. If this occurs, the controller terminates until the issue can be resolved.

c. Post Control Loop

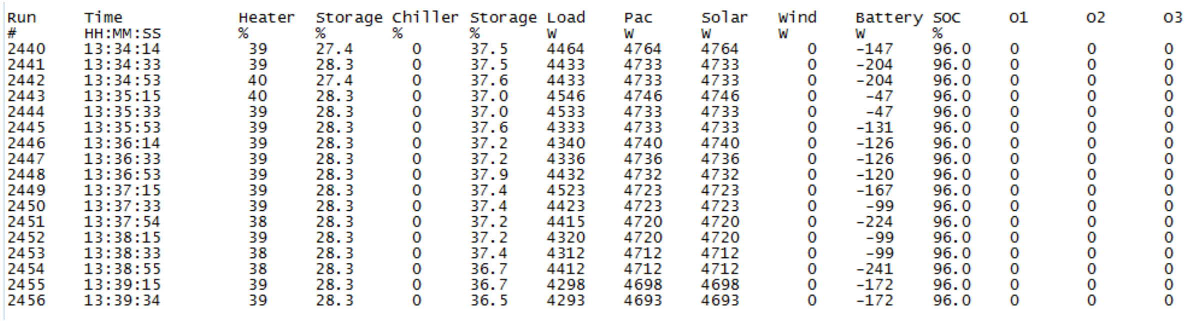
If the control loop terminates, this section will turn off the loads, close the data file, and close the Modbus port to the WebBox. This occurs if a condition was set to end the control loop, or if MATLAB encountered and error while running the control loop.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND ANALYSIS

The microgrid controller was developed and implemented in January 2016 using MATLAB software. It ran continuously, collecting data, through June 2016 when research concluded. Throughout this period, the controller was updated with improved controls and communications. The basic control strategy remained the same.

The controller was monitored from the desktop computer. Data was displayed to the screen after every iteration of the control loop. This allows the user to actively monitor values such as power generation, load demand, and energy storage levels. It also informed the user of controller performance adjusting load power to power generation. Every fifteen iterations (five minutes) of data was plotted to a chart. This figure allows the user to visually observe the performance of the controller from the previous six hours. Figures 21 and 22 are examples of the data displayed to the desktop and data plotted to the chart. Note that “Pac” is AC power generation.



```
Run     Time            Heater  Storage Chiller Storage Load    Pac     Solar   Wind    Battery SOC     O1      O2      O3
#       HH:MM:SS        %       %       %       %       W       W       W       W       W       %
2440    13:34:14         39     27.4       0    37.5    4464    4764    4764       0    -147    96.0    0       0       0
2441    13:34:33         39     28.3       0    37.5    4433    4733    4733       0    -204    96.0    0       0       0
2442    13:34:53         40     27.4       0    37.6    4433    4733    4733       0    -204    96.0    0       0       0
2443    13:35:15         40     28.3       0    37.0    4546    4746    4746       0     -47    96.0    0       0       0
2444    13:35:33         39     28.3       0    37.0    4533    4733    4733       0     -47    96.0    0       0       0
2445    13:35:53         39     28.3       0    37.6    4333    4733    4733       0    -131    96.0    0       0       0
2446    13:36:14         39     28.3       0    37.2    4340    4740    4740       0    -126    96.0    0       0       0
2447    13:36:33         39     28.3       0    37.2    4336    4736    4736       0    -126    96.0    0       0       0
2448    13:36:53         39     28.3       0    37.9    4432    4732    4732       0    -120    96.0    0       0       0
2449    13:37:15         39     28.3       0    37.4    4523    4723    4723       0    -167    96.0    0       0       0
2450    13:37:33         39     28.3       0    37.4    4423    4723    4723       0     -99    96.0    0       0       0
2451    13:37:54         38     28.3       0    37.2    4415    4720    4720       0    -224    96.0    0       0       0
2452    13:38:15         39     28.3       0    37.2    4320    4720    4720       0     -99    96.0    0       0       0
2453    13:38:33         38     28.3       0    37.4    4312    4712    4712       0     -99    96.0    0       0       0
2454    13:38:55         38     28.3       0    36.7    4412    4712    4712       0    -241    96.0    0       0       0
2455    13:39:15         39     28.3       0    36.7    4298    4698    4698       0    -172    96.0    0       0       0
2456    13:39:34         39     28.3       0    36.5    4293    4693    4693       0    -172    96.0    0       0       0
```

Figure 21. Typical Control Data Displayed to Desktop

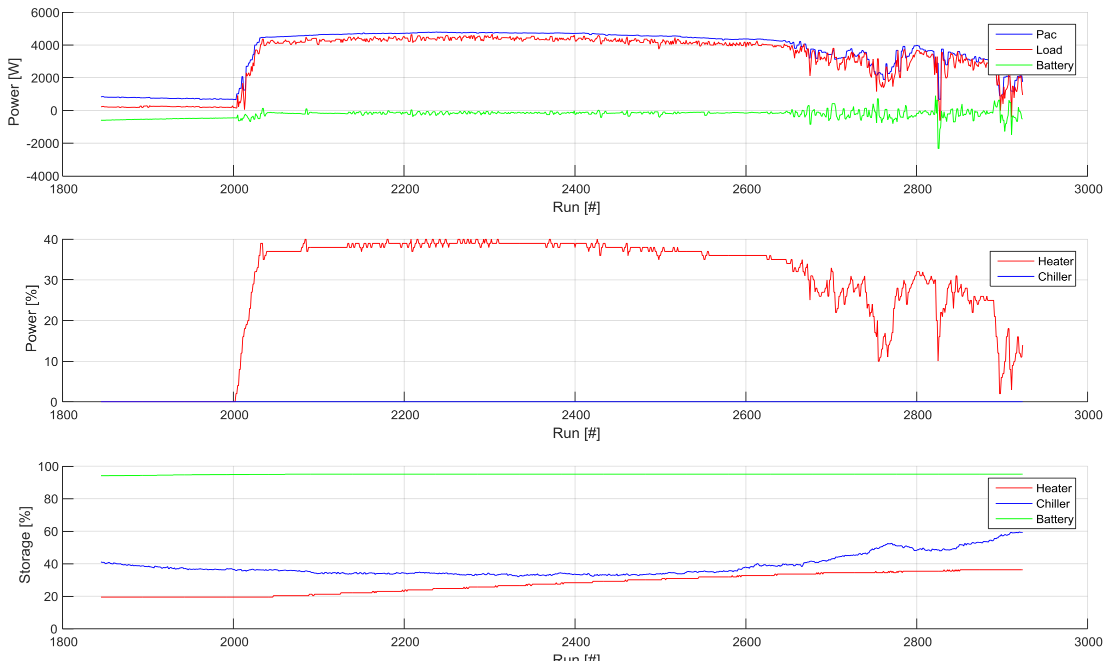


Figure 22. Control Data Plotted to Chart

The data was also recorded to a text file, which can be used for later analysis. This text file includes specific data from the system such as power levels of the loads, thermal storage levels and temperatures, metrics on the batteries, inverter inputs and outputs from the solar panels and wind turbines, as well as many other values. A custom Excel document was created as a template to easily plot the data. Once the data was imported, the template would plot the same data as the controller, as well as power data from the inverters, solar panels, and wind turbines. Figure 23 shows an example of data collected by the controller, and Figures 24–27 show examples of plots made by the Excel template. All plots display the data as a function of time throughout a single day.

CONTROL DATA																								
Time	Heater				Chiller					Load		Power Generation			Battery					Overrides		
	Power	Max	Temp	Storage	Power	Max	T_In	T_Out	Storage	Power	Power	Total	Solar	Wind	Vtg	Cur	Pwr	SOC	Mode			
HH:MM	%	%	F	%	%	%	F	F	%	W	W	W	W	W	V	A	W	%	-	1	2	3
9:00	2	50	450	27.9	0	100	58.3	57.0	88.3	459	474	874	874	0	53.2	-7.8	-415	89.1	1767	0	0	0
9:00	2	50	450	27.9	0	100	58.3	57.1	88.3	425	372	872	872	0	53.2	-8.4	-447	89.1	1767	0	0	0
9:01	3	50	450	27.9	0	100	58.3	57.1	88.5	415	362	862	862	0	53.2	-8.4	-447	89.1	1767	0	0	0
9:01	3	50	450	27.9	0	100	58.3	57.1	88.7	539	562	862	862	0	52.9	-6.1	-323	89.1	1767	0	0	0
9:01	3	50	450	27.9	0	100	58.3	57.1	88.5	544	567	867	867	0	52.9	-6.1	-323	89.1	1767	0	0	0
9:02	3	50	450	27.9	0	100	58.3	57.1	88.5	532	567	867	867	0	53.1	-6.3	-335	89.1	1767	0	0	0
9:02	3	50	450	27.9	0	100	58.3	57.1	88.5	593	628	928	928	0	53.1	-6.3	-335	89.1	1767	0	0	0
9:02	3	50	450	27.9	0	100	58.3	57.1	88.2	448	404	1004	1004	0	53.0	-10.5	-557	89.1	1767	0	0	0
9:03	5	50	450	27.9	0	100	58.3	57.1	88.4	559	615	1115	1115	0	53.0	-10.5	-557	89.1	1767	0	0	0
9:03	5	50	450	27.9	0	100	58.3	57.1	88.4	769	684	1184	1184	0	53.2	-7.8	-415	89.2	1767	0	0	0
9:04	5	50	450	27.9	0	100	58.3	57.1	88.2	784	699	1199	1199	0	53.2	-7.8	-415	89.2	1767	0	0	0
9:04	5	50	450	27.9	0	100	58.3	57.1	88.1	791	706	1206	1206	0	53.2	-7.8	-415	89.2	1767	0	0	0
9:04	5	50	450	27.9	0	100	58.3	57.1	87.9	745	635	1235	1235	0	53.3	-9.2	-490	89.2	1767	0	0	0
9:05	6	50	450	27.9	0	100	58.3	57.1	87.9	765	655	1255	1255	0	53.3	-9.2	-490	89.2	1767	0	0	0
9:05	6	50	450	27.9	0	100	58.3	57.1	87.7	922	855	1255	1255	0	52.9	-6.3	-333	89.2	1767	0	0	0
9:05	6	50	450	27.9	0	100	58.3	57.1	87.8	863	796	1196	1196	0	52.9	-6.3	-333	89.2	1767	0	0	0
9:06	6	50	450	27.9	0	100	58.3	57.1	87.4	862	859	1159	1159	0	53.0	-5.6	-297	89.2	1767	0	0	0
9:06	6	50	450	27.9	0	100	58.3	57.1	87.4	900	834	1134	1134	0	53.1	-4.4	-234	89.2	1767	0	0	0
9:06	6	50	450	27.9	0	100	58.3	57.1	87.7	877	811	1111	1111	0	53.1	-4.4	-234	89.2	1767	0	0	0
9:07	6	50	450	27.9	0	100	58.4	57.1	87.5	815	749	1049	1049	0	53.1	-4.4	-234	89.2	1767	0	0	0
9:07	6	50	450	27.9	0	100	58.4	57.1	87.2	891	749	1049	1049	0	52.7	-3.0	-158	89.2	1767	0	0	0
9:08	5	50	450	27.9	0	100	58.4	57.1	87.2	871	729	1029	1029	0	52.7	-3.0	-158	89.2	1767	0	0	0
9:08	5	50	450	27.9	0	100	58.4	57.1	87.2	765	729	1029	1029	0	52.8	-5.0	-264	89.2	1767	0	0	0
9:08	5	50	450	27.9	0	100	58.4	57.1	87.2	773	737	1037	1037	0	52.8	-5.0	-264	89.2	1767	0	0	0
9:09	5	50	450	27.9	0	100	58.4	57.1	87.2	778	742	1042	1042	0	52.8	-5.0	-264	89.2	1767	0	0	0
9:09	5	50	450	27.9	0	100	58.4	57.1	87.2	834	709	1009	1009	0	52.9	-3.3	-175	89.2	1767	0	0	0
9:09	4	50	450	27.9	0	100	58.4	57.1	87.2	814	689	989	989	0	52.9	-3.3	-175	89.2	1767	0	0	0
9:10	4	50	450	27.9	0	100	58.4	57.1	87.1	814	689	989	989	0	52.9	-3.3	-175	89.2	1767	0	0	0

Figure 23. Control Data Section of Excel Template

Figure 23 shows the data collected by the controller that is used by the control loop. This includes power levels of the heater and chiller, thermal storage levels and temperatures, load demand, power generation, battery metrics, and the status of the three override controls. This data is used to generate the plots in Figure 24. The remaining data collected by the controller that is used to generate plots in Figure 25–27 is not shown.

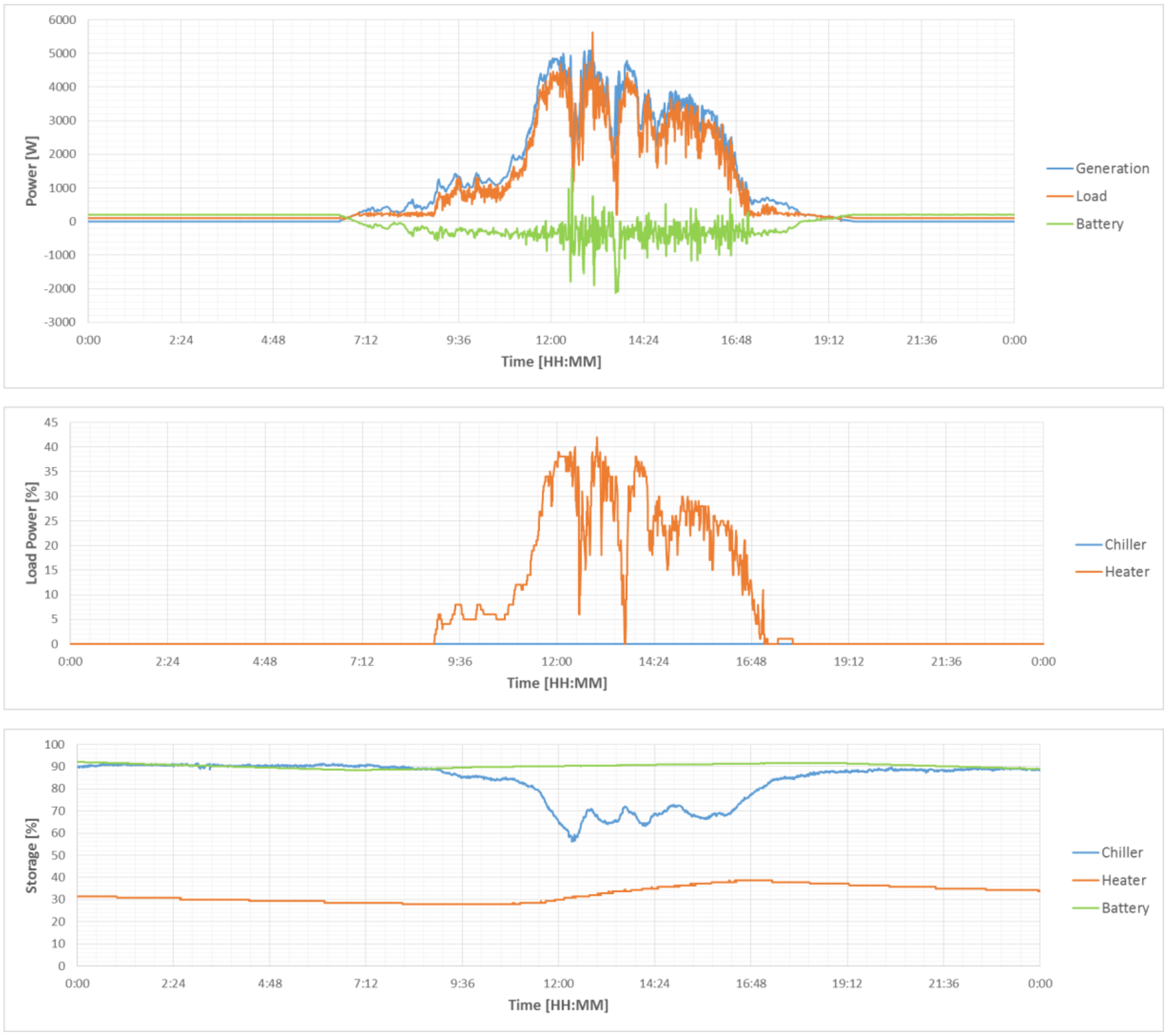


Figure 24. Control Data Plots in Excel Template

Figure 24 shows the plots generated using the control data. The top plot shows generation, load, and battery powers in Watts. This plot shows the performance of the controller and how closely it is matching load demand to power generation. Negative battery power indicates power supplied to the batteries for charging, and positive battery power indicates power supplied from the batteries to the loads. The middle plot shows load powers in percentage. These powers are determined by the controller. The bottom plot shows thermal and electrical storage levels in percentage. These are used by the controller to determine how to prioritize loads.

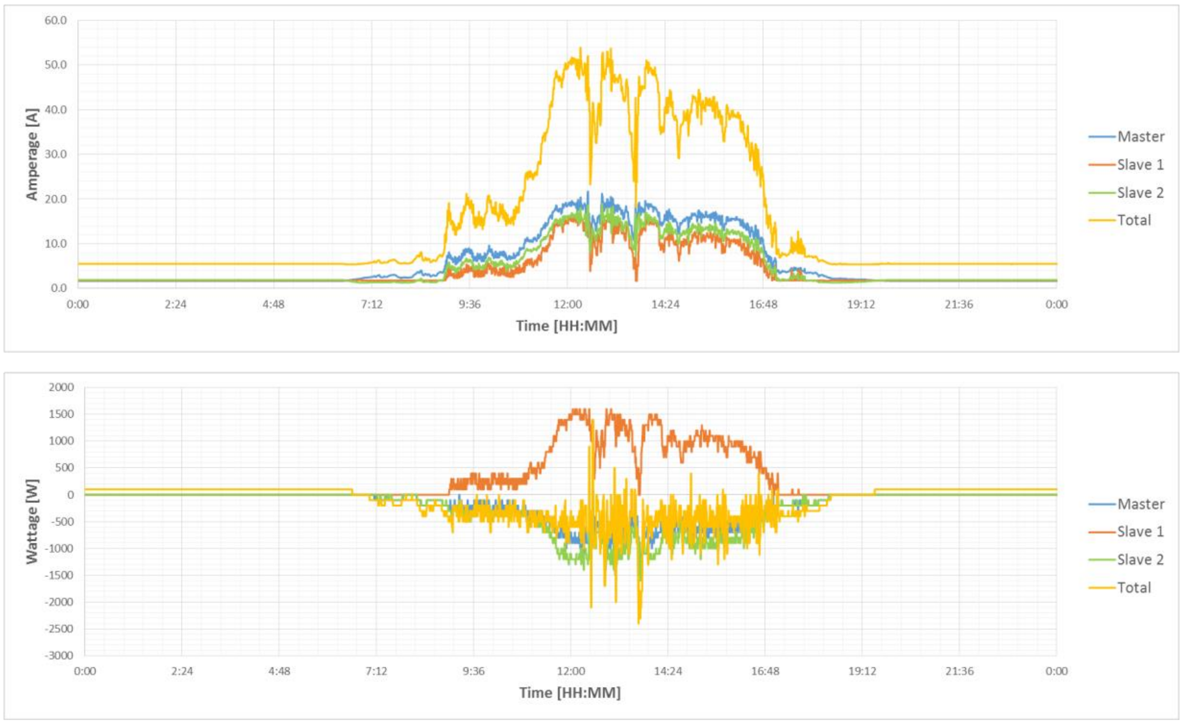


Figure 25. Inverter Data Plots in Excel Template

Figure 25 shows the plots generated using the inverter data from the Sunny Islands. The three Sunny Island inverters each regulate one of the three phases of AC power. The top plot shows inverter currents in amperage. The bottom plot shows inverter power in watts. Both plots include a total value, which is the sum of the three inverters.

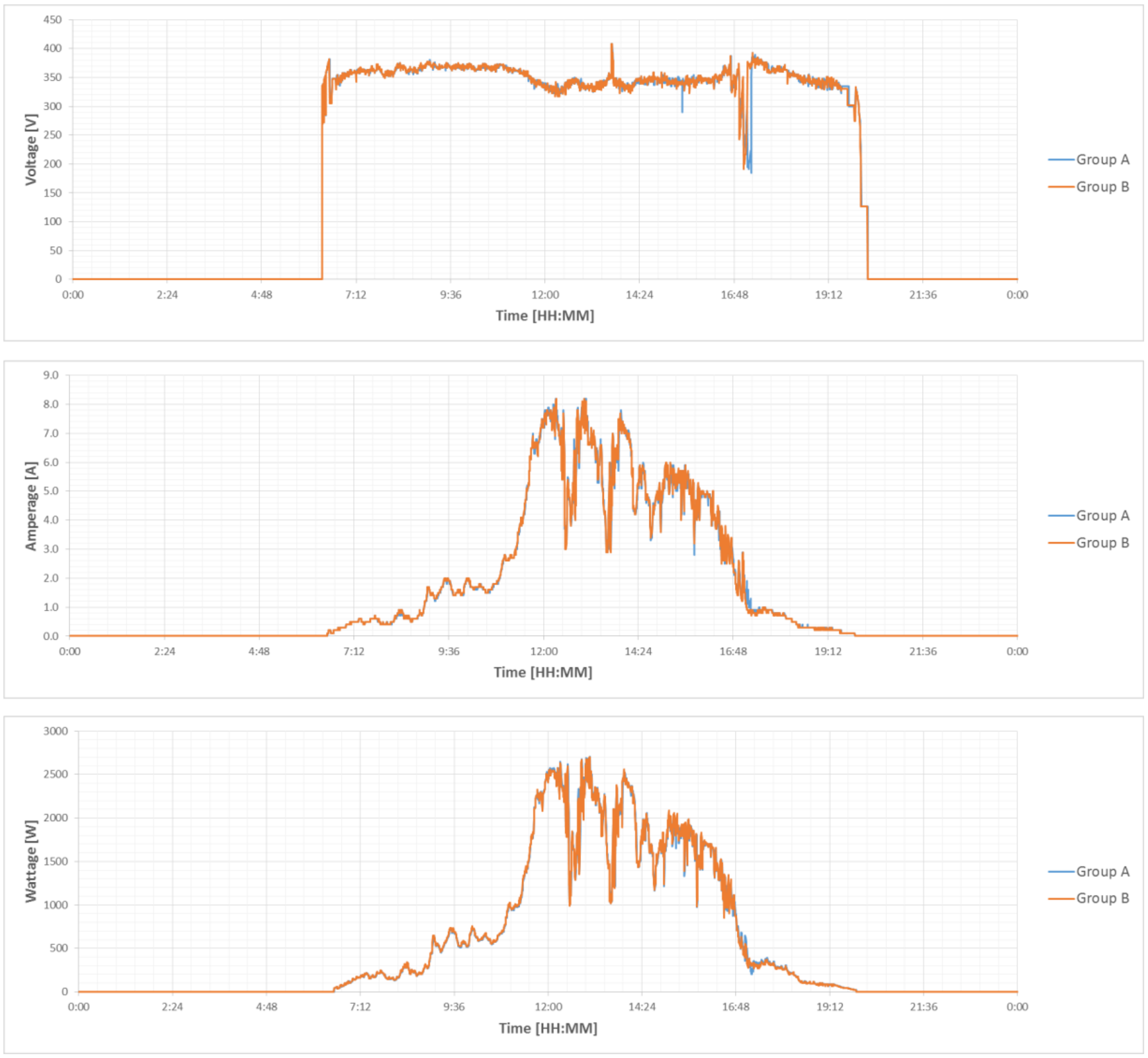


Figure 26. Solar Inverter Data Plots in Excel Template

Figure 26 shows the plots generated using the solar inverter data. The Sunny Boy solar inverter receives separate DC inputs from the two groups of solar panels, group A and B. The top plot shows DC voltage in volts, the middle plot shows DC current in amps, and the bottom plot shows DC power in Watts. On this particular day, the two groups of solar panels produced nearly identical power profiles.

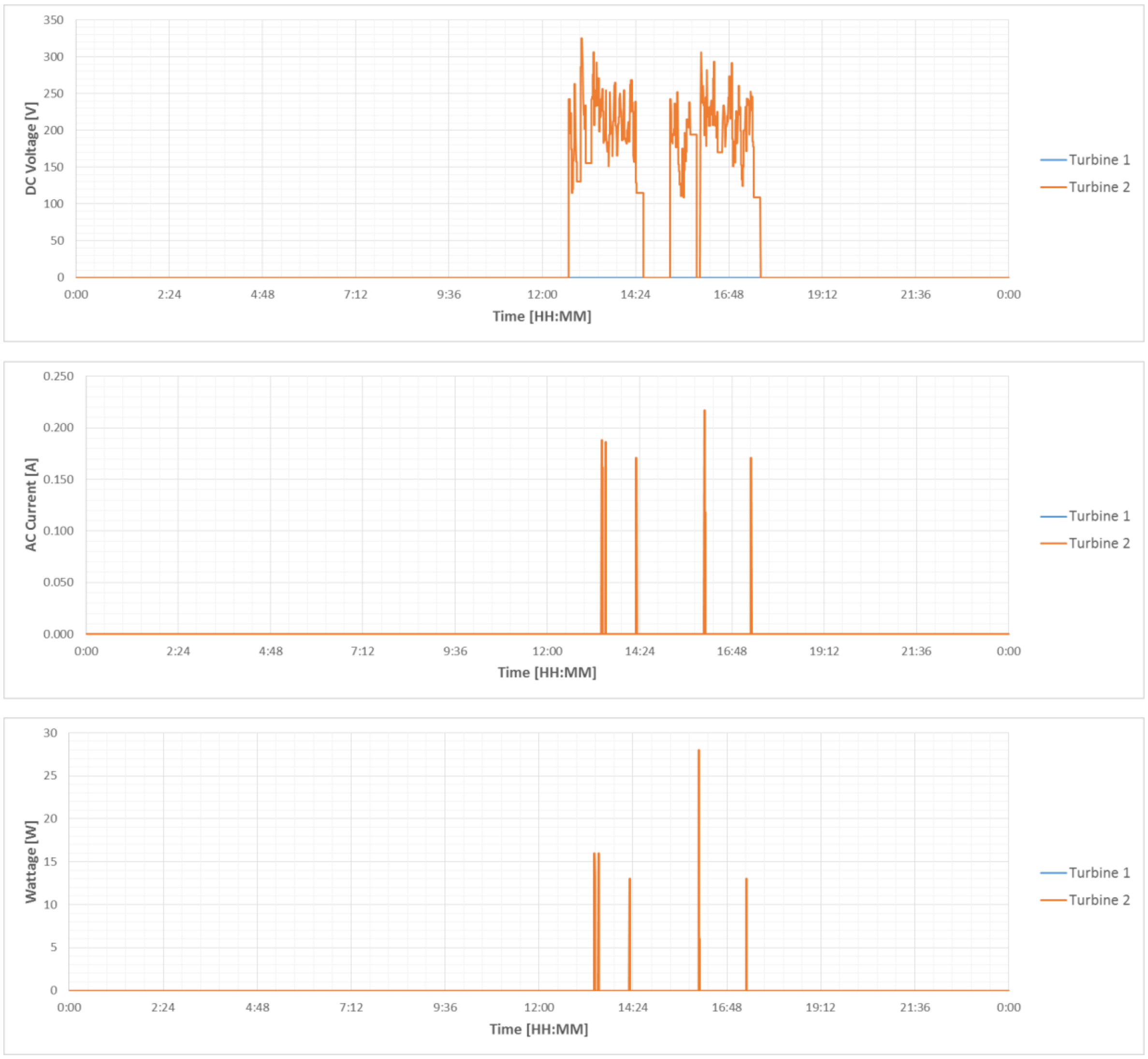


Figure 27. Wind Inverters Data in Excel Template

Figure 27 shows the plots generated using the wind inverters data. The two Sunny Boy wind inverters convert variable DC power from the Aurora WBIs to AC power. The top plot shows the DC voltage inputs to the inverters in volts, the middle plot shows AC current output from the inverters in amps, and the bottom plot shows AC power output in Watts. On this particular day only one wind turbine was generating power.

It should be noted that the Sunny Boy inverters do not accept all of the DC power inputs from the Aurora inverters. This can been seen in the difference between the DC inputs in the top plot of Figure 26 and the AC outputs in the middle and bottom plots.

This is caused by a mismatch in operational voltage range between the two inverters. The Aurora Wind Box Inverter (WBI) outputs DC power from 50–530V to the SMA Sunny Boy inverter [30]. The voltage increases with increasing wind speeds and power generation by the wind turbines. However, the SMA Sunny Island inverter only accepts DC inputs from 250–480V [31]. The minimum DC start voltage is 250V. This is summarized in Table 2. DC voltage inputs below 250V and above 480V cannot be synchronized to the grid. This causes the usable power generation from the wind turbines to decrease drastically. The solution would be to replace the SMA Sunny Boy inverters with inverters that can operate from 50–530V DC.

Table 2. DC Operating Voltage of Wind Turbine Inverters

Inverter	Aurora WBI	SMA Sunny Boy
DC Voltage Range	50 - 530	250 - 480

A. LOAD POWER PROFILES

The first step in developing the microgrid controller was to determine the power profiles of all of the loads. This was needed to correlate load power percentage with actual load power in watts. The controller used this correlation to adjust load demand by adjusting the load power percentage. Figure 28 shows the power profiles of the loads.

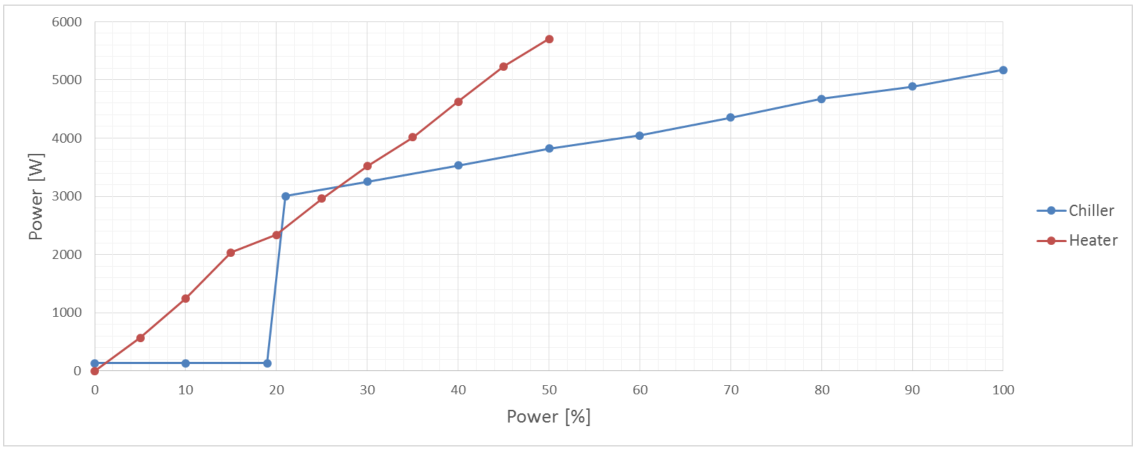


Figure 28. Power Profiles of the Loads

The large startup power required of the chiller can be seen in Figure 28. It takes approximately 3kW to run the chiller at its lowest power setting, which is 20%. The chiller power percentage correlates to the compressor speed. The initial startup power of the chiller is even greater than 3kW, due to the spike in power when turning on the compressor. Running the chiller at 100% power requires slightly more than 5kW. The heater power percentage correlates to the amount of power sent to the electrical heater elements inside the heater. Since there is no minimum power associated with heater elements, the heater can run at very low power. The maximum power percentage of the heater is limited by the power generation resources. The power generation of the solar panels peaks around 6kW on an average sunny day. Since it takes nearly 6kW to run the large heater at 50% power, the maximum power of the heater is set to 50%.

B. CONTROL DATA FOR COLD THERMAL STORAGE

The controller was first tested with only the cold thermal storage system. Figure 29 shows data collected by the controller on January 26, 2016. At this point, the controller could only poll data about the microgrid inverters from the WebBox webpage and send voltage commands to the chiller. Polling data from the webpage meant that each iteration of the control loop took a minimum of two minutes. This particularly day was also mostly cloudy, which caused solar power generation to be low. The wind turbines produced little or no power.

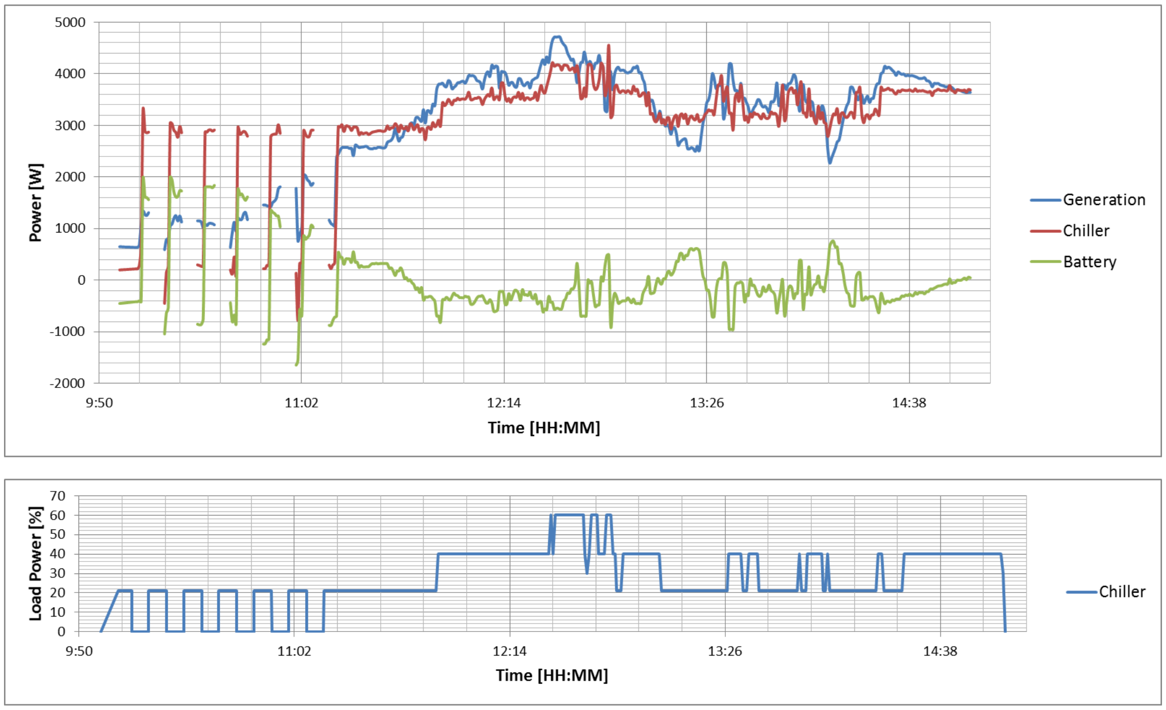


Figure 29. Control Data Running Chiller

The top plot of Figure 28 shows power generation, chiller power, and battery power in watts from approximately 0950 to 1500. The bottom plot shows the chiller power percentage over the same period. The objective of the control strategy is to match the chiller power to the power generation, without drawing power from the batteries. The large startup power of the chiller, the low power generation, and the long control loop iterations all caused the performance of the controller to be relatively poor on this particular day. The lower power standby control caused the chiller to cycle on and off seven times from approximately 1000 to 1100. The control cycled the chiller on to determine available power generation. Once it was determined there was insufficient power, the control cycled the chiller off. Each time the control cycled the chiller, large amounts of power were pulled from the batteries.

The chiller eventually turned on and remained running at 1120. However, the chiller continued drawing power from the batteries for another 20 minutes, until power generation increased. As power generation continued to increase, the controller attempted to increase power to the chiller, and match the chiller load to power generation. Around

1230, the chiller reached a maximum power for the day of 60%. At 1500, the chiller turned off for the day. The relative performance of the controller was considered poor since power was often drawn from the batteries to provide power to the load.

C. CONTROL DATA FOR HOT THERMAL STORAGE

Once communications with the heater was established using the serial-USB COM port adapter, the heater was incorporated into the controller. The control strategy was tested again running the heater as the only load. Figure 30 shows data collected by the controller on April 30, 2016 from 0600 until 1424. This particular day was mostly sunny with minimal power generation by the wind turbines.

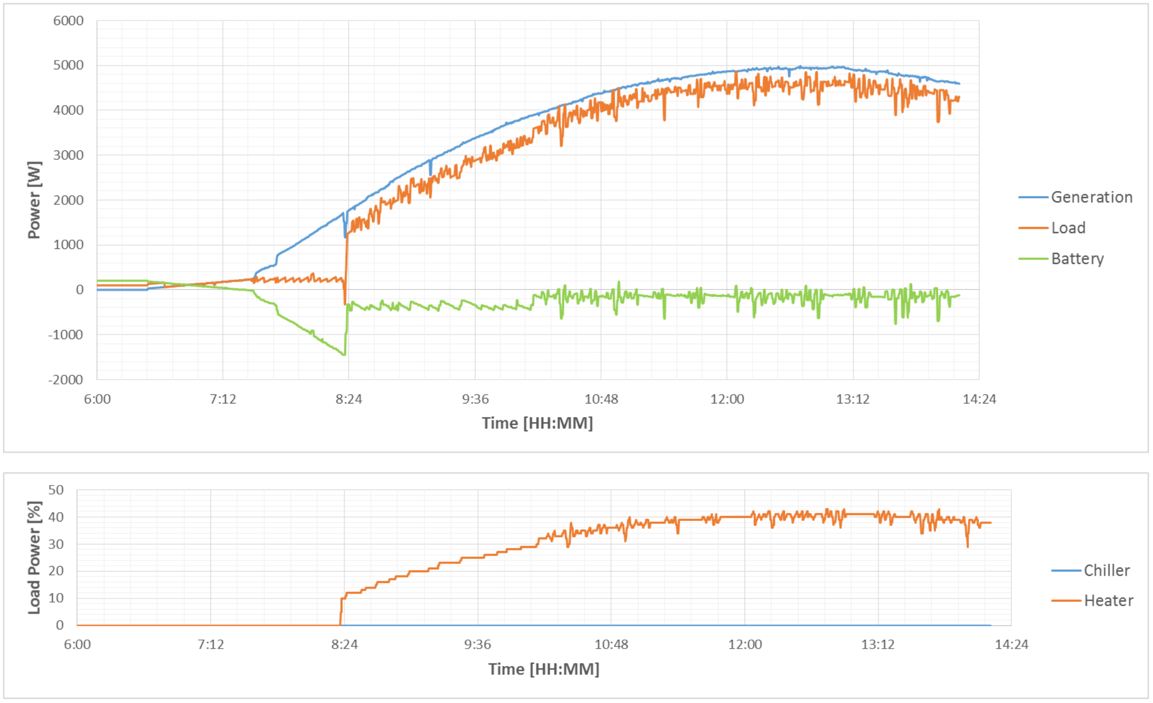


Figure 30. Control Data Running Heater

Figure 30 shows nearly ideal performance by the controller. In the morning, the max power point tracking control allowed the batteries to charge. This control prevented any loads from running until the power to the batteries had peaked at 0821. At 0823 there was a dip in power generation, which indicated that power to the batteries had peaked and the heater load was introduced. From 0823 until 1012 the controller operated under normal conditions. The controller increased power to the heater as power generation increased, while allowing 200–400W of power to the batteries.

At 1012 the batteries were charged to a point that the high battery SOC control took effect. This control decreased the amount of power to the batteries to 100–200W. This is can be observed in the figures as the load power approaches power generation and the power to the batteries approaches zero. Performance of the controller was considered nearly ideal. Taking into consideration the power to the batteries, the controller was able to match load demand to within 5% below and 1% above power generation. The controller was manually terminated at 1412 to implement updates.

D. CONTROL DATA FOR COMBINED THERMAL STORAGE

Previous testing demonstrated the control strategy and developed controller worked successfully with individual loads. The next step was to update the controller to run both the heater and the chiller simultaneously. This presented a unique challenge in that the controller needed to prioritize loads. Thermal storage percent of completion (POC) was ultimately used to prioritize the loads. POC was determined by the controller based on current storage levels as well as the desired storage levels set by the user. Figure 31 shows control data collected while running the updated controller on May 14, 2016. The bottom plot of this figure includes the storage levels for the batteries, ice storage tank, and heater. It should be noted that the desired storage level of ice was set higher than the heater by the user. The performance of the controller on this day is representative of an average day. The weather was partly cloudy with no power generation from the wind turbines.

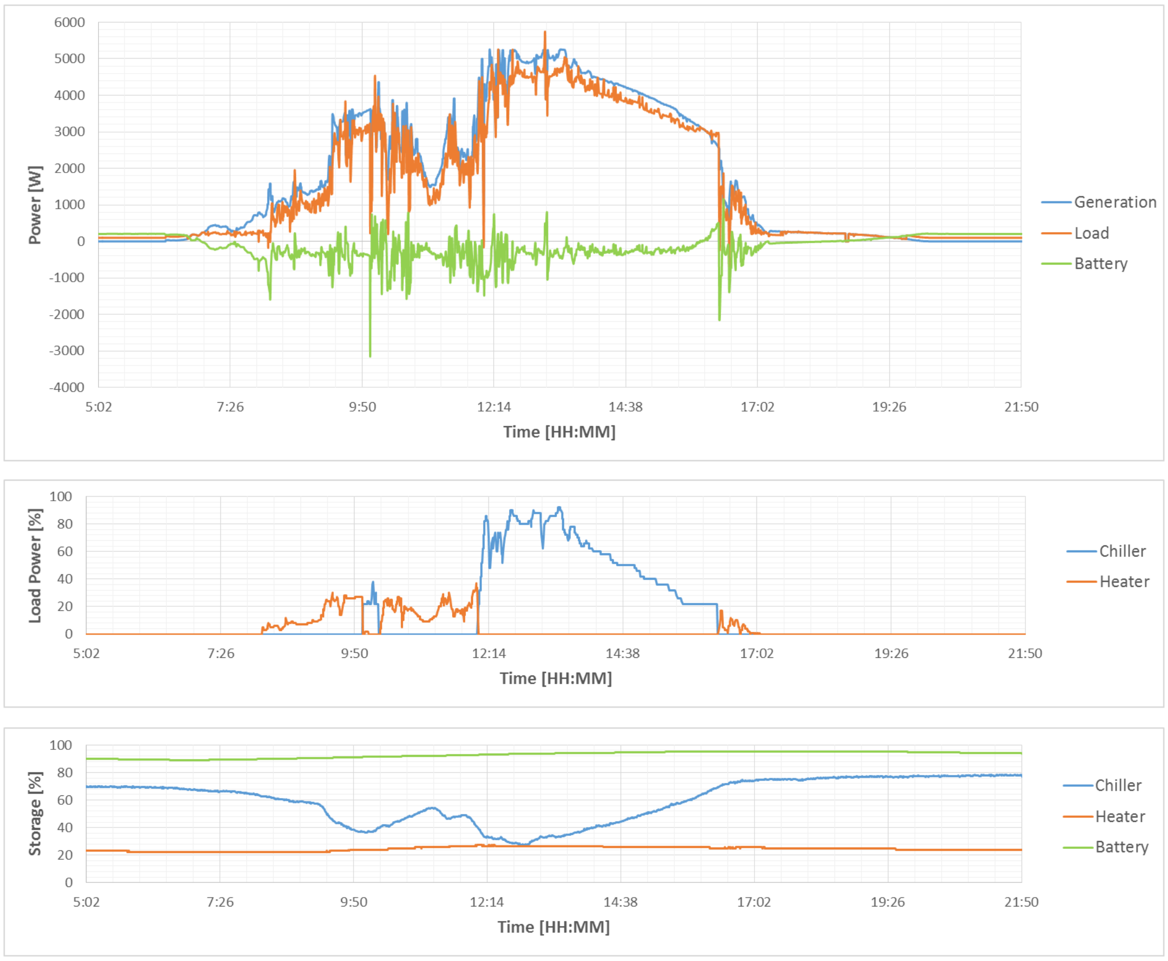


Figure 31. Control Data Running Both Loads

The following analysis of Figure 31 is done chronologically to explain the controller operation. In the morning, the max power point control allowed the batteries to charge until 0809, when the loads were introduced. The heater storage levels were much lower than the ice storage levels in the morning, so the controller prioritized the heater. Additionally, as average power generation was below 3kW, the controller would default to the heater regardless. The heater was run until 0958. At this point, the ice storage levels began to drop, causing the controller to prioritize the chiller. The controller ran the chiller from 0959 until 1015 with the heater turned off. While ice levels did increase, the chiller was shut off because it was drawing power from the batteries.

From 1016 until 1201, the controller defaulted to running the heater because average power generation remained below 3kW. At 1203, dropping ice levels and increased power generation caused the controller to once again prioritize the chiller. Chiller power increased to a max of 92% at 1329 and continued to run until 1619. During this time, ice levels increased from 41.2 to 68.4%. At 1620, insufficient power generation caused the controller to default to the heater for the remainder of the day. Overall the performance of the controller on this day was relatively good, considering it was partly cloudy. Power was drawn from the batteries to cover periods when load demand briefly exceeded power generation, but the controller was able to quickly compensate the load power. Battery levels peaked at 95%, which indicates a decent charge level for the day. It was also shown that the heater worked well in combination with the chiller. If power generation is too low to run the chiller, the controller defaults to the heater. Running the heater allows the controller to determine the available power generation. This eliminates the need for the low power standby control.

Figure 32 shows another example of control data collected on May 22, 2016. This particular day was mostly sunny. The controller never prioritized the chiller as ice storage levels never dipped below heater storage levels, and user preference for storage levels was set to the same value. The batteries entered a boost charge cycle at 0048. This enabled override three, preventing any loads from running until the boost charge completed at 1108. The controller normally would have turned on the loads at 0845, when power the batteries peaked. The interpolated maximum potential power generation is shown by the dashed lined. The area between the dashed line and the solid line for power generation represents energy that could have potentially been available to run the loads, if there had been demand. The objective of the control strategy is to prevent unused energy such as this. Theoretically, loads could have been introduced to utilize the excess power. However, that would have diverted power from the batteries so it is considered acceptable in this case in order to improve the lifetime of the batteries.

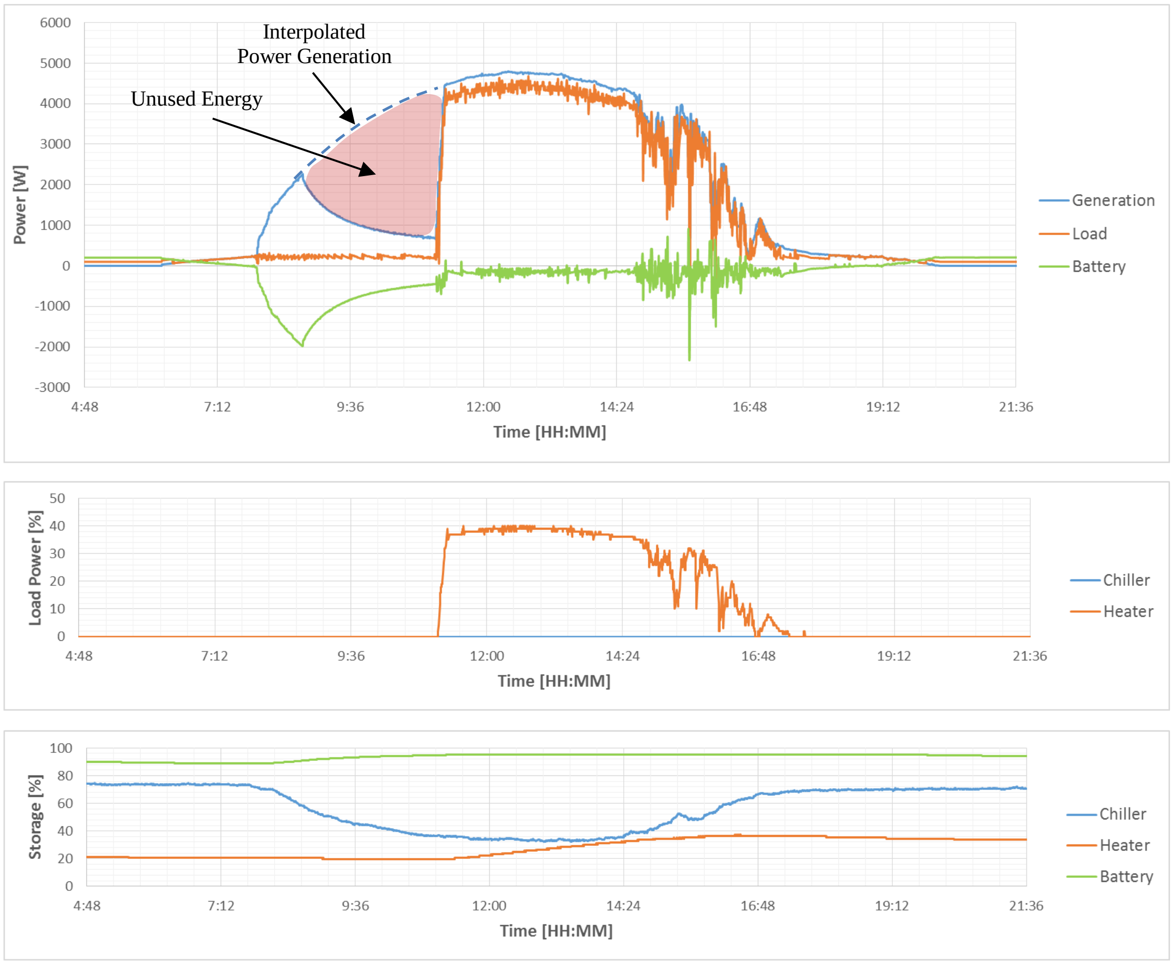


Figure 32. Control Data with Battery Boost Charge

Based on the results collected over this six-month period, the desired control strategy was successfully implemented. The microgrid controller was able to match load demand to power generation, without requiring the batteries to run the loads. The batteries were only used to cover brief periods when load demand exceeds power generation and to stabilize the grid at night when there was no power generation. MATLAB allowed for easily implementation and modification of the microgrid controller. Modification made to the controller throughout this period improved the performance of the controller. However, the basic control strategy remained the same.

E. LONG-TERM STORAGE LEVELS

The ultimate purpose of this microgrid system is to provide heating and cooling applications, while also covering minor electrical loads. The storage devices were used as a buffer to store excess energy when it was not needed, and to provide energy when it was needed. Therefore, the ability of the thermal storage devices as well as the batteries to retain energy is very important. Energy lost by the storage devices increases the inefficiency of the system. Figure 33 shows the energy storage profiles of the batteries and the thermal storage devices over a three-day period.

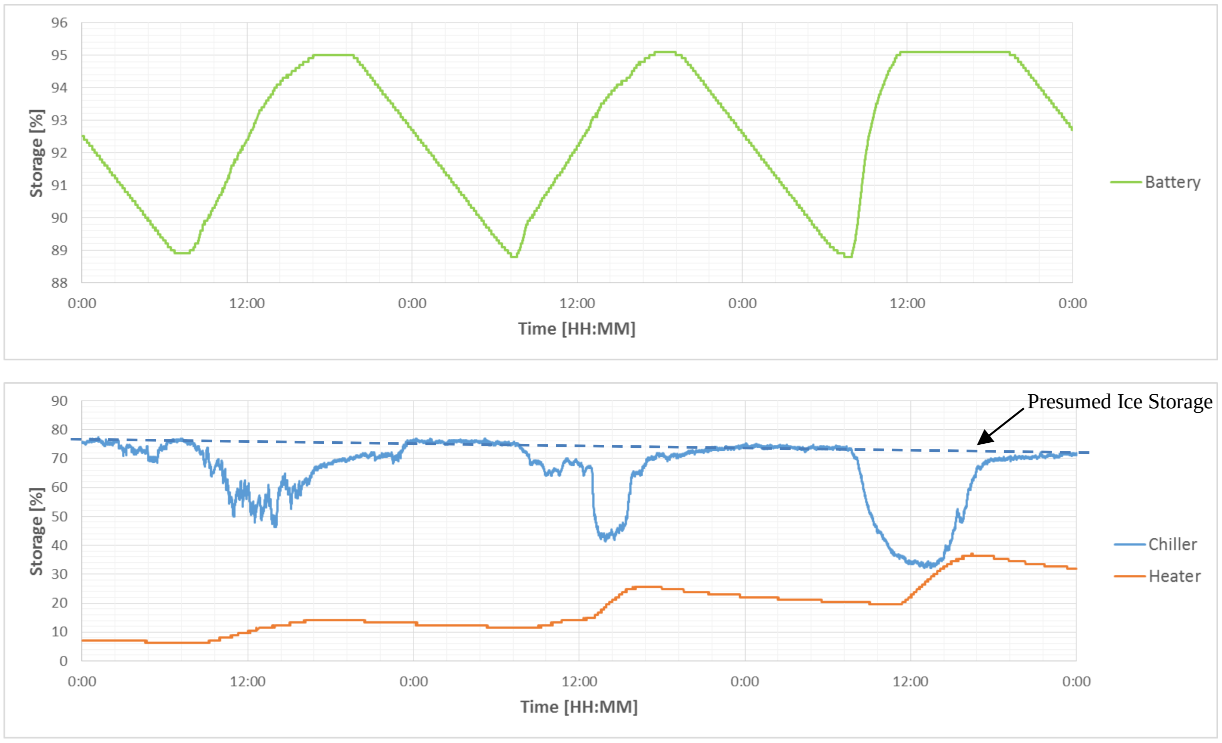


Figure 33. Storage Levels over Three Days

The top plot of Figure 33 shows the battery SOC. The battery SOC peaks in the afternoon, right before sunset, and decreases overnight. The batteries lose charge overnight mainly from stabilizing the microgrid. Minor electric loads cause the microgrid to have a minimum of 200W of load. Overnight, the batteries typically lose 6% of charge, which is approximately equivalent to 5kWh. The first two days show a typical charge profile and the third day shows a boost charge profile.

The thermal storage levels decreased only due to heat loses. The heater had much greater heat loses than the ice storage, due to the larger temperature gradients. The heat loss by the heater was also dependent on storage level, having greater heat losses at higher storage levels due to higher temperatures. This made it difficult to maintain the heater at high storage levels. Since the ice tank used the phase change of water to stores energy, which occurs at a constant temperate, its heat loses were independent of storage levels as the temperature gradient is constant. The heater typically lost 2–8% of its storage levels overnight, depending on temperature. This is equivalent to 2.4-9.6kWh of energy.

The ice storage levels were difficult to measure accurately. The ice storage level sensor tends to indicate lower storage levels during the day, and higher storage levels at night. This is likely because the ice storage tank is located outside, in direct sunlight. Since the sensor is on the top of the tank, as the sun warms the top of the tank, the sensor falsely indicates lower storage levels. The chiller was never run during the three day period shown in Figure 32. The dashed line illustrates what is believed to be the actual trend in storage level. If the dip in storage levels during the day is ignored, the ice storage tank lost approximately 2–4% storage each day. This is equivalent to 3.4-6.8kWh of energy. Based on these results, summarized in Table 3, the thermal storage devices are not recommend for long-term storage.

Table 3. Overnight Losses in Storage Levels

Storage Device	**Batteries**	**IceBank**	**Heater**
Overnight Losses (kWh)	5	3.4 - 6.8	2.4 - 9.6

THIS PAGE INTENTIONALLY LEFT BLANK

V. FUTURE STUDY

This section discusses potential follow on research for this project. Recommendations on minor improvements to the current systems are covered in the following section. These areas of research are intended to improve the overall performance of the microgrid system. Two possible areas of research are discussed for future study, integrating weather forecasting into the controller and performing an exergy analysis on the overall system.

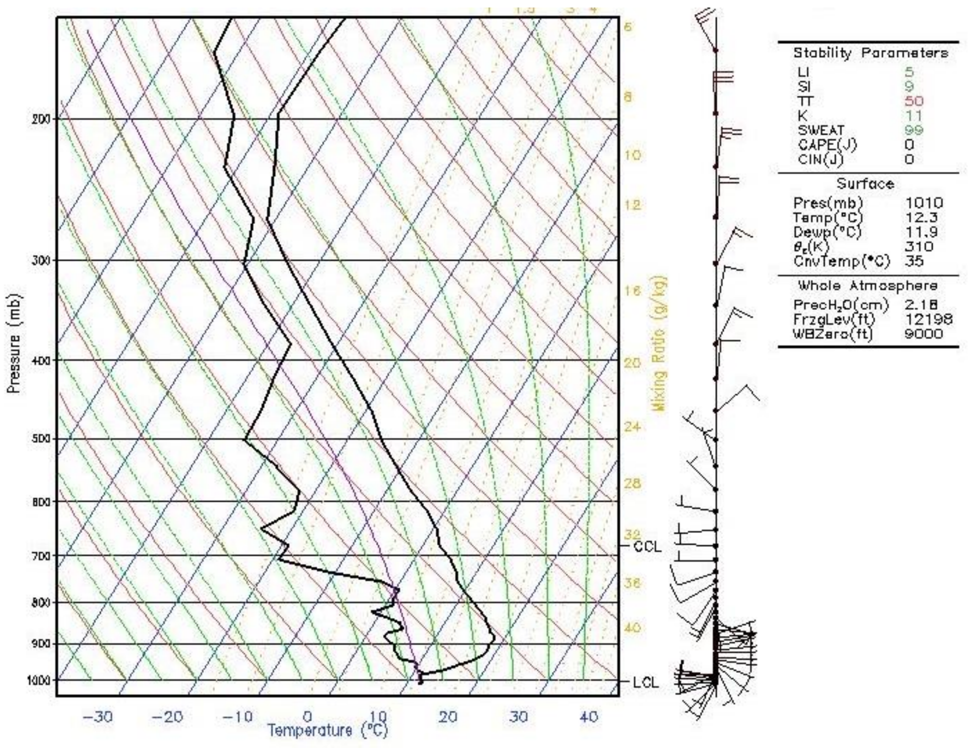
A. WEATHER FORECASTING

Integrating weather forecasting and climatology would introduce an element of predictability into the controller. Weather forecasts predict the short-term weather conditions. Climatology predicts the long-term weather conditions using the average of numerous factors over the past thirty years. Since the weather conditions affect the performance of this system, the controller would be able to anticipate these changes and make adjustments to the controls.

Weather conditions would affect two different areas of the systems performance, power generation and thermal storage levels. Weather conditions have the greatest impact on power generation. Power is generated using solely wind and solar energy. Power generation by the wind turbines is directly related to wind speeds. Power generation by the solar panels is directly related to cloud coverage and time of year. Based on forecasted wind speeds and cloud coverage, the controller can predict future levels of power generation. This could greatly improve the performance of the controller since it currently only responds to changes in power generation. It may also alter the strategy of the controller. For instance, if weather forecasts indicated high potential power generation, the controller could draw power from electrical storage. Excess power generation later on would be used to recharge the electrical storage. Conversely, if conditions indicated low potential power generation, the controller would store more energy in electrical storage. This would ensure sufficient electrical power to support the system in the long run.

Thermal storage levels are affected by weather conditions since it impacts the demands for heating and cooling. Based on certain forecast conditions, such as temperature, humidity, and precipitation, the controller can predict if there will be a greater demand for hot or cold thermal storage. The controller can then prioritize that load in advance, ensuring that there is sufficient thermal energy to meet demand.

Successful implementation of predictability based on weather requires the ability for accurate weather forecasting. The Fleet Numerical Meteorology and Oceanography Center (FNMOC) uses high-performance supercomputers to run meteorological models from their operations center in Monterey, CA. Operated by the U.S. Navy, FNMOC provides worldwide meteorology and oceanography support to U.S. and coalition forces [32]. For this research, FNMOC provided unclassified weather forecasting using the Coupled Ocean Atmosphere Mesoscale Prediction System (COAMPS) run on their beta servers. The COAMPS forecast uses synoptic models to predict 72 hour weather conditions and probabilistic models to predict conditions from 72 hours to two weeks. Forecasts are generated for the exact location of the IMPREL building. Figure 33 shows a Skew-T Log-P thermodynamic diagram generated by the COAMPS model. This diagram illustrates weather conditions at the lab location. Figure 34 illustrates cloud coverage and wind speed generated by the COAMPS model for the general region. The model uses a 555-meter resolution.



Stability Parameters	
LI	5
SI	9
TT	50
K	11
SWEAT	99
CAPE(J)	0
CIN(J)	0
Surface	
Pres(mb)	1010
Temp(°C)	12.3
Dewp(°C)	11.9
θ_E(K)	310
CnvTemp(°C)	35
Whole Atmosphere	
PrecH_2O(cm)	2.18
FrzgLev(ft)	12198
WBZero(ft)	9000

Figure 34. COAMPS Skew-T Log-P for IMPREL Location

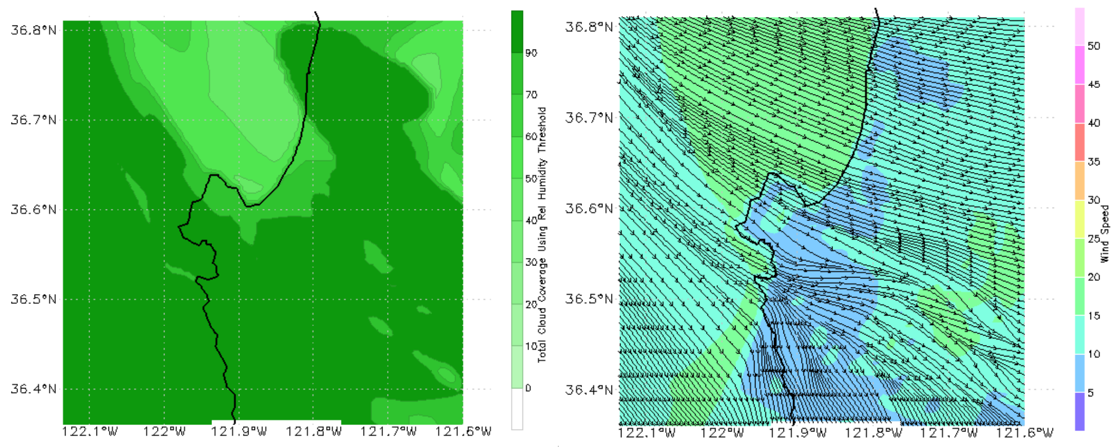


Figure 35. COAMPS Weather Forecast—Cloud Coverage (left)
Wind Speeds (right)

In most situations, predictability can be integrated into the microgrid controller using local weather forecasts. However, certain DOD assets such as forward operating bases would not otherwise have access to weather forecasts. FNMOC is uniquely suited to provide weather models to worldwide DOD assets that may potentially use this microgrid system.

B. EXERGY ANALYSIS

Exergy is a thermodynamic property used to measure the maximum useful work done by process. An exergy analysis is often performed in engineering in order to use energy more efficiently. It can be used to optimize the processes and equipment of a system. Since this system has already been designed and tested, an exergy analysis can be performed to show how effectively it is using the available energy. This analysis would provide a starting comparison point for traditional systems.

One particular area an exergy analysis would be useful is in the comparison of thermal storage conversions to electrical storage conversions, shown in Figure 35. Since electrical storage devices use DC power, energy has to pass through the inverters twice, once when it is being sent to the device and again when the device feeds power back to the microgrid. Energy is lost each time it passes through the inverter. Since the thermal storage is not converted back to electricity, theoretically it does not experience as much loses.

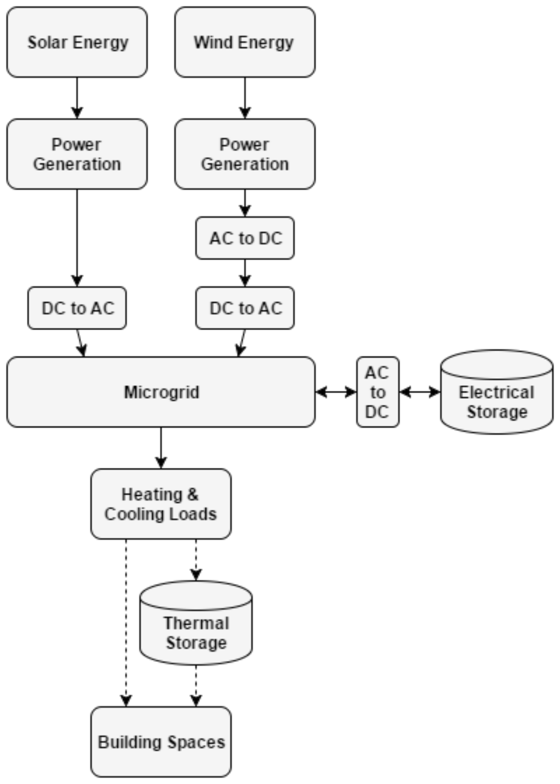


Figure 36. Energy Paths for Electrical and Thermal Storage Devices

VI. RECOMMENDATIONS

There are multiple improvements that can be made to both the controller and to the microgrid system that would improve the overall performance. The follow list describes recommendations for the microgrid controller. They are listed with what is believed to be the easiest implemented recommendations first.

- Contact SMA to resolve issues with Modbus communications. Modbus returns an "illegal data address" when polling the register addresses of the second DC inputs to the solar Sunny Boy. The battery SOC register in the Sunny Island only includes two significant digits, versus three displayed by the webpage. The inverters will also stop communicating over Modbus for a period of a few minutes to a few hours. This has only happened a few times, mostly with the solar Sunny Boy, and is seemingly random.

- Contract Trane to allow the UC400 controller to accept BACnet write commands. Trane originally disabled this function when installing the chiller. Since the controller can now execute BACnet communications, it has the ability to send BACnet commands to the controller. If the compressor speed could be regulated using BACnet command it would eliminate the need for the NI cDAQ device.

- Create a Simulink model of the controller using blow flow diagram. Simulink provides a graphical editor that could be used to model the controller and implement more advanced controls. This would also allow for better visualization of how the control works.

- Replace the MATLAB controller with an industrial controller. Although MATLAB allows for easy design and implementation, it does not have the redundancies or backups to run a system like this continuously. An industrial controller would allow the system to operate autonomously for long periods of time.

The following list describes recommendations for the microgrid system. This mainly includes improvements to equipment. They are listed with what is believed to be the easiest implemented recommendations first.

- Install the air ventilation system with the IceBank so it can be used to provide air conditioning to the lab spaces. The main obstacle to installing the ventilation is the fact that the IceBank is located outside. Therefore, ventilation would likely have to be run through the outer wall of the building.

- Install different DC to AC inverters that can accept the entire range of DC voltage outputs from the Aurora WBI’s. Installing new inverters is required in order to utilize any meaningful power generation from the wind turbines.

- Install a supercapacitor bank for electrical storage. Supercapacitors are advantageous over batteries since they can be repeated cycled without deterioration or completely drained without damage. Since electrical storage is only used to stabilize the grid, and not run loads directly, large amounts of energy storage is not required. Supercapacitors can be used alongside the current battery bank or can replace the battery bank altogether.

VII. CONCLUSIONS

This research was the final step completing the IMPREL microgrid system. The components were integrated into system and a controller was designed and implemented. The most challenging part of this research was designed the communications network. This was required in order to implement a centralized controller that could communicate with all components of the system, across multiple architectures. Data collected over a six month period demonstrated that the microgrid system could successfully match load demand to within 5% below and 1% above power generation by utilizing thermal storage under normal conditions. Improvements were continuously made to the controller, but the control strategy remained the same.

All of the equipment worked as intended, however, a mismatch between the aurora inverter and the SMA inverters caused dramatically poor performance by the wind turbines. The solar panels provided nearly all of the power generation used during this research. The ability to finely adjust load power was a vital part of implementing the control strategy. The controller required this ability to match the demand from the heater and chiller to renewable power generation. The chiller was more difficult to operate due to its large startup power. The controller was able to mitigate this issue when operating both loads by running only the heater at low power. The controller would only switch to running the chiller when there was sufficient power generation. The storage devices lost an expected amount of energy during long term operations. The amount of energy lost by the thermal storage devices was slightly greater than the electrical storage, with the greatest losses by the heater due to the high internal temperatures.

The demonstrated microgrid system has many potential applications. Using commercially available equipment makes this system easy to modify and install. The system can be redesigned using the multi-physics methodology, based on variations in end-use energy. Using the control strategy of matching load demand to power generation also allows for more effective use of renewable energy. As shown in the results, the control strategy used all available power from the renewable resources. Compared to the traditional approach of sizing the renewable resources to match peak load demand, this strategy requires fewer renewable resources to meet the same demand, reducing size, cost, and the amount of unused resources. Further analysis can be done to accurately determine the overall performance of this system. The IMPREL microgrid system utilizing the developed microgrid controller was a successful demonstration. This research proves the end-use energy design concept proposed by the multi-physics methodology.

APPENDIX A. MATLAB CODE

A. MICROGRID LOAD CONTROLLER

```
%----------------------------------------------------------------------
----
%------------------------ MICROGRID CONTROLLER -------------------------
----
%----------------------------------------------------------------------
----
close all; clear; clc;
%----------------------------------------------------------------------
----
%---------------------------- USER INPUTS ------------------------------
----
%----------------------------------------------------------------------
----
% Specify Heater - Large Heater = COM4(4), Small Heater = COM3(3)
COM = 4;
FanSetting = 'OFF';
% Set Thermal Storage Percentage (0-100)
Heater_Storage_Setpoint = 90;
Ice_Storage_Setpoint = 90;
% Set Start and Stop Hour (0-24)
Start_Time = 7;
Stop_Time = 18;
% Set Minimum Power Requirements (Watts)
MPP_Startup = 600;
Low_Power_Standby = 200;
Low_Power_Shutdown = 200;
% Set Standby Time (min)
Standby_Time = 15;
% Set Battery SOC Limits
BatSoc_Lower_Limit = 85;
%----------------------------------------------------------------------
----
%----------------------------------------------------------------------
----

% Establish Modbus Connection with SMA Sunny Webbox
IPaddress = '192.168.0.168';
Port = 502;
WebboxTCP = tcpip(IPaddress, Port);
set(WebboxTCP,'InputBufferSize', 512);
WebboxTCP.ByteOrder = 'bigEndian';
fopen(WebboxTCP);

% Establish Address Registry for Trane BACnet Devices
BACnet('AddressCache');

% Define URLs
% url1 =
'http://192.168.0.168/plant_current.htm?DevKey=SI6048UH:1260019920&DevC
lass=Sunny%20Island';
% url4 =
'http://192.168.0.168/plant_current.htm?DevKey=WRTU1O73:1913091742&DevC
lass=Sunny%20Island';

% Create Data File
Control_Data = fopen(['Control_Data_', date, '.txt'],'w');
fprintf(Control_Data,'Time \t Heater \t Max \t Temp \t Storage \t
Chiller \t Max \t T_In \t T_Out \t Storage \t Load \t Load \t Pac \t
Solar \t Wind \t BatVtg \t BatCur \t BatPower \t SOC \t Mode \tO1 \t O2
\t O3 \t\t Time \t Fac \t Vac \t Iac \t Pwr \t Vac \t Iac \t Pwr \t Vac
\t Iac \t Pwr \t TotInvCur \t TotInvPwrAt \t TotInvPwrRT \t\t Time \t
Panels_A \t\t\t Panels_B \t\t\t Grid_A \t\t\t Grid_B \t\t\t Total \t\t
Time \t WT1_DC \t\t WT1_AC \t\t\t\t\t Mode \t WT2_DC \t\t WT2_AC
\t\t\t\t\t\t Mode \t Total \r\n');
fprintf(Control_Data,'HHMMSS \t P \t P \t F \t %% \t %% \t %% \t F \t F
\t %% \t W \t W \t W \t W \t W \t V \t A \t W \t %% \t - \t - \t - \t -
\t\t HHMMSS \t Hz \t V \t A \t W \t V \t A \t W \t v \t A \t W \t A \t
W \t W \t\t HHMMSS \t V \t A \t W \t V \t A \t W \t V \t A \t W \t V \t
A \t W \t W \t\t HHMMSS \t V \t A \t Hz \t V \t V \t A \t W \t - \t V
\t A \t Hz \t V \t V \t A \t W \t - \t W \r\n');

% Preallocate Memory
Heater_Pwr    = zeros(1,10000);
Heater_Pwr_Max = zeros(1,10000);
Heater_Temp    = zeros(1,10000);
Heater_Storage = zeros(1,10000);

Chiller_Pwr    = zeros(1,10000);
Chiller_Pwr_Max = zeros(1,10000);
Chiller_T_In   = zeros(1,10000);
Chiller_T_Out   = zeros(1,10000);
Ice_Storage    = zeros(1,10000);

Pac_Load     = zeros(1,10000);
TotInvPwrAt    = zeros(1,10000);
InvPwrAt     = zeros(1,10000);
InvPwrAtSlv1   = zeros(1,10000);
InvPwrAtSlv2   = zeros(1,10000);
InvVtg      = zeros(1,10000);
InvVtgSlv1    = zeros(1,10000);
InvVtgSlv2    = zeros(1,10000);
TotInvCur     = zeros(1,10000);
InvCur      = zeros(1,10000);
InvCurSlv1    = zeros(1,10000);
InvCurSlv2    = zeros(1,10000);
InvFrq      = zeros(1,10000);
TotInvPwrRt    = zeros(1,10000);
TotBatCur     = zeros(1,10000);
BatSoc      = zeros(1,10000);
BatVtg      = zeros(1,10000);
BatChrgOp     = zeros(1,10000);
Mode_WT1     = zeros(1,10000);
```

```
Ipv_WT1      = zeros(1,10000);
Vpv_WT1      = zeros(1,10000);
Pac_WT1      = zeros(1,10000);
Vac_L1_WT1    = zeros(1,10000);
Vac_L2_WT1    = zeros(1,10000);
Iac_WT1      = zeros(1,10000);
Fac_WT1      = zeros(1,10000);
Mode_WT2     = zeros(1,10000);
Ipv_WT2      = zeros(1,10000);
Vpv_WT2      = zeros(1,10000);
Pac_WT2      = zeros(1,10000);
Vac_L1_WT2    = zeros(1,10000);
Vac_L2_WT2    = zeros(1,10000);
Iac_WT2      = zeros(1,10000);
Fac_WT2      = zeros(1,10000);
AMsAmp      = zeros(1,10000);
AMsVol      = zeros(1,10000);
AMsWatt      = zeros(1,10000);
Pac_PV      = zeros(1,10000);
BMsAmp      = zeros(1,10000);
BMsVol      = zeros(1,10000);
BMsWatt      = zeros(1,10000);
GridMsWphsA    = zeros(1,10000);
GridMsWphsB    = zeros(1,10000);
GridMsVphsA    = zeros(1,10000);
GridMsVphsB    = zeros(1,10000);
GridMsAphsA    = zeros(1,10000);
GridMsAphsB    = zeros(1,10000);
Pac_WT      = zeros(1,10000);
Pac        = zeros(1,10000);
BatPower     = zeros(1,10000);
Load       = zeros(1,10000);
Pac_ave      = zeros(1,10000);
BatPower_ave   = zeros(1,10000);
t         = zeros(1,10000);
tplot       = zeros(1,10000);

% Turn OFF Overrides
Override_1 = zeros(1,10000);
Override_2 = zeros(1,10000);
Override_3 = zeros(1,10000);
% NOTE - Override 1 is ON by default once control loop resets

% Print Header to Screen
fprintf('Run \tTime  \tHeater\tStorage\tChiller\tStorage\tLoad\tPac
\tSolar\tWind\tBattery\tSOC \tO1\tO2\tO3 \n');
fprintf('#  \tHH:MM:SS \t%%  \t%%  \t%%  \t%%  \tW  \tW  \tW  \tW  \tW
\t%% \n');

%------------------------------------------------------------------------
----
%----------------------------- CONTROL LOOP -----------------------------
----
%------------------------------------------------------------------------
----
```

```
Run = 1;
i = 1;
while Run == 1
  try

  % Time
  tic
  t(i) = now;
  tplot(i) = str2double(datestr(now,'HHMMSS'));
  time = fix(clock);
  hour = time(1,4);
  minute = time(1,5);

  % Plant Parameters (UnitID = 2)
  Pac_Load(i)    = Modbus(2,30775,WebboxTCP);
  % Sunny Island (UnitID = 3)
  TotInvPwrAt(i)  = Modbus(3,30775,WebboxTCP);
  InvPwrAt(i)    = Modbus(3,30777,WebboxTCP);
  InvPwrAtSlv1(i)  = Modbus(3,30779,WebboxTCP);
  InvPwrAtSlv2(i)  = Modbus(3,30781,WebboxTCP);
  InvVtg(i)     = Modbus(3,30783,WebboxTCP)/100;
  InvVtgSlv1(i)   = Modbus(3,30785,WebboxTCP)/100;
  InvVtgSlv2(i)   = Modbus(3,30787,WebboxTCP)/100;
  TotInvCur(i)   = Modbus(3,30795,WebboxTCP)/1000;
  InvCur(i)     = Modbus(3,30797,WebboxTCP)/1000;
  InvCurSlv1(i)   = Modbus(3,30799,WebboxTCP)/1000;
  InvCurSlv2(i)   = Modbus(3,30801,WebboxTCP)/1000;
  InvFrq(i)     = Modbus(3,30803,WebboxTCP)/100;
  TotInvPwrRt(i)  = Modbus(3,30807,WebboxTCP);
  TotBatCur(i)   = Modbus(3,30843,WebboxTCP)/1000;
  BatSoc(i)     = Modbus(3,30845,WebboxTCP);
  BatVtg(i)     = Modbus(3,30851,WebboxTCP)/100;
  BatChrgOp(i)   = Modbus(3,30853,WebboxTCP);
%   AptTmBoost    = Modbus(3,40039,WebboxTCP);
  % Wind Turbine 1 Inverter (UnitID = 5)
  Mode_WT1(i)    = Modbus(5,30241,WebboxTCP);
  Ipv_WT1(i)    = Modbus(5,30769,WebboxTCP)/1000;
  Vpv_WT1(i)    = Modbus(5,30771,WebboxTCP)/100;
  Pac_WT1(i)    = Modbus(5,30775,WebboxTCP);
  Vac_L1_WT1(i)   = Modbus(5,30783,WebboxTCP)/100;
  Vac_L2_WT1(i)   = Modbus(5,30785,WebboxTCP)/100;
  Iac_WT1(i)    = Modbus(5,30797,WebboxTCP)/1000;
  Fac_WT1(i)    = Modbus(5,30803,WebboxTCP)/100;
  % Wind Turbine 2 Inverter (UnitID = 4)
  Mode_WT2(i)    = Modbus(4,30241,WebboxTCP);
  Ipv_WT2(i)    = Modbus(4,30769,WebboxTCP)/1000;
  Vpv_WT2(i)    = Modbus(4,30771,WebboxTCP)/100;
  Pac_WT2(i)    = Modbus(4,30775,WebboxTCP);
  Vac_L1_WT2(i)   = Modbus(4,30783,WebboxTCP)/100;
  Vac_L2_WT2(i)   = Modbus(4,30785,WebboxTCP)/100;
  Iac_WT2(i)    = Modbus(4,30797,WebboxTCP)/1000;
  Fac_WT2(i)    = Modbus(4,30803,WebboxTCP)/100;
  % Solar Inverter (UnitID = 6)
  AMsAmp(i)     = Modbus(6,30769,WebboxTCP)/1000;
  AMsVol(i)     = Modbus(6,30771,WebboxTCP)/100;
  AMsWatt(i)    = Modbus(6,30773,WebboxTCP);
  Pac_PV(i)     = Modbus(6,30775,WebboxTCP);
%   BMsAmp(i)     = Modbus(6,30957,WebboxTCP)/1000;
%   BMsVol(i)     = Modbus(6,30959,WebboxTCP)/100;
%   BMsWatt(i)    = Modbus(6,30961,WebboxTCP);
  GridMsWphsA(i) = Modbus(6,30777,WebboxTCP);
  GridMsWphsB(i) = Modbus(6,30779,WebboxTCP);
  GridMsVphsA(i) = Modbus(6,30783,WebboxTCP)/100;
  GridMsVphsB(i) = Modbus(6,30785,WebboxTCP)/100;
  GridMsAphsA(i) = Modbus(6,30797,WebboxTCP)/1000;
  GridMsAphsB(i) = Modbus(6,30799,WebboxTCP)/1000;

  % Calculated Variables
  BatPower(i)    = TotBatCur(i)*BatVtg(i);
  Pac_WT(i)     = Pac_WT1(i) + Pac_WT2(i);
  Pac(i)      = Pac_PV(i) + Pac_WT(i);
  Load(i)      = Pac(i) + BatPower(i);

  % These values cannot be accessed using Modbus - use urlfilter
%   BatSoc(i)     = urlfilter(url1,'BatSoc');
%   BMsAmp(i)     = urlfilter(url4,'B.Ms.Amp');
%   if BMsAmp(i) == 5
%     BMsAmp(i) = 0;
%   end
%   BMsVol(i)     = urlfilter(url4,'B.Ms.Vol');
%   if BMsVol(i) == 6
%     BMsVol(i) = 0;
%   end
%   BMsWatt(i)    = urlfilter(url4,'B.Ms.Watt');
%   if BMsWatt(i) == 7
%     BMsWatt(i) = 0;
%   end

  % Chiller Storage Level and Temperatures
  [~,Ice_Storage(i)]    = BACnet('Read',10000,0,9,85);
  [~,Chiller_T_In(i)]    = BACnet('Read',10000,0,5,85);
  [~,Chiller_T_Out(i)]   = BACnet('Read',10000,0,4,85);

  % Heater Storage Level
  if COM == 3
    Heater_Temp_Max = 1100;
  elseif COM == 4
    Heater_Temp_Max = 1200;
  end
  if i == 1
    [AvgTemp,~] = Heater_Comm(COM,0,'OFF');
    Heater_Storage(1) = ((AvgTemp - 70)/(Heater_Temp_Max - 70))*100;
  else
    Heater_Storage(i) = ((Heater_Temp(i-1) - 70)/(Heater_Temp_Max -
70))*100;
  end

  % Power Moving Averages
  if i >= 6
```

```
    Pac_ave(i) = mean(Pac((i-5):i));
    BatPower_ave(i) = mean(BatPower((i-5):i));
  end

  % Max Power Point Tracking (Override 1)
  MPP = max(Pac_PV);
  if hour >= Start_Time
    if i > 1 && Override_1(i-1) == 1
      if MPP < MPP_Startup || Pac_PV(i) > 0.75*MPP
        Override_1(i) = 1;
      end
    end
  else
    Override_1(i) = 1;
  end

  % Low Power Shutdown (1/2) (Override 2)
  if hour >= Start_Time && hour < Stop_Time
    if i > 6 && mod(i,5) == 0
      if Override_1(i) == 0
        if Pac_ave(i) < Low_Power_Shutdown
          Override_2(i:i+4) = 1;
        end
      end
    end
  end

  % Low Power Standby (1/2) (Override 2)
  if hour >= Start_Time && hour < Stop_Time;
    if mod(i,Standby_Time*3 + 15) == 0
      if Override_1(i) == 0
        if Heater_Pwr(i) > 0 || Chiller_Pwr(i) > 0
          if Pac_ave(i) < Low_Power_Standby && Pac_ave(i) >
Low_Power_Shutdown
            Override_2(i:(i+(Standby_Time*3-1))) = 1;
          end
        end
      end
    end
  end

  % Battery SOC Warning (Override 3)
  if BatSoc(i) < BatSoc_Lower_Limit
    Heater_Comm(COM,0,'OFF');
    Chiller_Comm(0);
    if i > 1
      if Override_3(i-1) == 0 || mod(i,5) == 0
        fprintf('\n')
        warning('Battery SOC %2.1f%% \n', BatSoc(i))
      end
    end
    Override_3(i:10000) = 1;
  elseif BatSoc(i) >= (BatSoc_Lower_Limit + 1)
    Override_3(i) = 0;
  end
```

```
% Battery Boost Charge (Override 3)
if BatChrgOp(i) == 1767
  if i > 1
    if Override_3(i-1) == 0 || mod(i,5) == 0
      fprintf('\n OVERRIDE: Battery in Boost \n\n')
      %6.0f [HHMMSS] Remaining \n\n', AptTmBoost)
    end
  end
  Override_3(i) = 1;
end

% Determine Percent of Completion (POC)
Heater_POC = Heater_Storage(i)/Heater_Storage_Setpoint*100;
if Heater_Storage_Setpoint == 0
  Heater_POC = 100;
end
Ice_POC = Ice_Storage(i)/Ice_Storage_Setpoint*100;
if Ice_Storage_Setpoint == 0
  Ice_POC = 100;
end

% Prioritize Load
if Heater_POC < Ice_POC
  Load_Priority = 'Heater';
else
  Load_Priority = 'Chiller';
end
if Pac_ave(i) < 3000
  if i > 9 && max(Chiller_Pwr(i-8:i)) > 0
  else
    Load_Priority = 'Heater';
  end
end

% Max Power Control
if COM == 4
  if Heater_POC > 80
    Heater_Pwr_Max(i+1) = 40;
  elseif Heater_POC > 90
    Heater_Pwr_Max(i+1) = 30;
  elseif Heater_POC >= 100
    Heater_Pwr_Max(i+1) = 0;
  else
    Heater_Pwr_Max(i+1) = 50;
  end
elseif COM == 3
  if Heater_POC > 80
    Heater_Pwr_Max(i+1) = 90;
  elseif Heater_POC > 90
    Heater_Pwr_Max(i+1) = 80;
  elseif Heater_POC >= 100
    Heater_Pwr_Max(i+1) = 0;
  else
    Heater_Pwr_Max(i+1) = 100;
  end
end
if Ice_POC > 80
  Chiller_Pwr_Max(i+1) = 75;
elseif Ice_POC > 90
  Chiller_Pwr_Max(i+1) = 50;
elseif Ice_POC >= 100
  Chiller_Pwr_Max(i+1) = 0;
else
  Chiller_Pwr_Max(i+1) = 100;
end

% Battery Power Control
if BatPower(i) < -800
  Load_Power = 5;
elseif BatPower(i) < -600
  Load_Power = 3;
elseif BatPower(i) < -500
  Load_Power = 2;
elseif BatPower(i) < -400
  Load_Power = 1;
elseif BatPower(i) < -200
  Load_Power = 0;
elseif BatPower(i) < -100
  Load_Power = - 1;
elseif BatPower(i) < 0
  Load_Power = - 2;
elseif BatPower(i) < 100
  Load_Power = - 3;
elseif BatPower(i) < 300
  Load_Power = - 4;
elseif BatPower(i) < 500
  Load_Power = - 5;
elseif BatPower(i) < 1000
  Load_Power = - 10;
elseif BatPower(i) < 1500
  Load_Power = - 15;
elseif BatPower(i) < 2000
  Load_Power = - 25;
else
  Load_Power = - 50;
end

% Low Power Startup
if Load_Power <= 0 && Heater_Pwr(i) == 0 && Chiller_Pwr(i) == 0
  if BatPower(i) < -150
    Load_Power = 1;
  end
end

% High Battery SOC Control
if BatSoc(i) >= 98 && Load_Power >= -10
  Load_Power = Load_Power + 2;
elseif BatSoc(i) >= 94 && Load_Power >= -5
  Load_Power = Load_Power + 1;
  end

  % Low Battery Power Control
  if BatPower_ave(i) > -150 && BatSoc(i) < 94
    Load_Power = Load_Power - 1;
  end

  % Decrease Power to Non-Prioritized Load
  if Load_Power <= 0
    if strcmp(Load_Priority,'Heater') == 1 && Chiller_Pwr(i) > 0
      Load_Priority = 'Chiller';
      if Heater_Pwr(i) < Heater_Pwr_Max(i)
        Load_Power = Load_Power - 10;
      end
    elseif strcmp(Load_Priority,'Chiller') == 1 && Heater_Pwr(i) > 0
      Load_Priority = 'Heater';
      if Chiller_Pwr(i) < Chiller_Pwr_Max(i)
        Load_Power = Load_Power - 10;
      end
    end
  end

  % Increase Power to Non-Prioritized Load
  if i > 1 && Load_Power > 0
    if strcmp(Load_Priority,'Heater') == 1 && Heater_Pwr(i) ==
Heater_Pwr_Max(i)
      Load_Priority = 'Chiller';
    elseif strcmp(Load_Priority,'Chiller') == 1 && Chiller_Pwr(i) ==
Chiller_Pwr_Max(i)
      Load_Priority = 'Heater';
    end
  end

  % Adjust Prioritized Load
  if strcmp(Load_Priority,'Heater') == 1
    Heater_Pwr(i+1) = Heater_Pwr(i) + Load_Power;
  elseif strcmp(Load_Priority,'Heater') == 1 && COM == 3
    Heater_Pwr(i+1) = Heater_Pwr(i) + 5*Load_Power;
  elseif strcmp(Load_Priority,'Chiller') == 1
    Chiller_Pwr(i+1) = Chiller_Pwr(i) + 2*Load_Power;
  end

  % Overshoot Control
  if i > 1
    if Heater_Pwr(i+1) > Heater_Pwr(i) && Heater_Pwr(i) > Heater_Pwr(i-
1)
      Heater_Pwr(i+1) = Heater_Pwr(i);
    elseif Heater_Pwr(i+1) < Heater_Pwr(i) && Heater_Pwr(i) <
Heater_Pwr(i-1)
      Heater_Pwr(i+1) = Heater_Pwr(i);
    elseif Chiller_Pwr(i+1) > Chiller_Pwr(i) && Chiller_Pwr(i) >
Chiller_Pwr(i-1)
      Chiller_Pwr(i+1) = Chiller_Pwr(i);
    elseif Chiller_Pwr(i+1) < Chiller_Pwr(i) && Chiller_Pwr(i) <
Chiller_Pwr(i-1)
      Chiller_Pwr(i+1) = Chiller_Pwr(i);
    end
  end

  % Heater Max and Min Power Control
  if Heater_Pwr(i+1) >= Heater_Pwr_Max(i+1);
    Heater_Pwr(i+1) = Heater_Pwr_Max(i+1);
  elseif Heater_Pwr(i+1) <= 0;
    Heater_Pwr(i+1) = 0;
  end

  % Chiller Max and Min Power Control
  if Chiller_Pwr(i+1) >= Chiller_Pwr_Max(i+1);
    Chiller_Pwr(i+1) = Chiller_Pwr_Max(i+1);
  elseif Chiller_Pwr(i+1) <= 22 && Chiller_Pwr(i) == 0
    Chiller_Pwr(i+1) = 22;
  elseif Chiller_Pwr(i+1) <= 22 && Chiller_Pwr(i+1) > 10
    Chiller_Pwr(i+1) = 22;
  elseif Chiller_Pwr(i+1) <= 10
    Chiller_Pwr(i+1) = 0;
  end

  % Overrides - Set Load Power to Zero
  if Override_1(i) == 1 || Override_2(i) == 1 || Override_3(i) == 1
    Heater_Comm(COM,0,'OFF');
    Chiller_Comm(0);
    Heater_Pwr(i:i+1) = 0;
    Chiller_Pwr(i:i+1) = 0;
  end

  % Overnight Standby (1/2) - Set Load Power to Zero
  if hour < Start_Time || hour >= Stop_Time
    Heater_Comm(COM,0,'OFF');
    Chiller_Comm(0);
    Heater_Pwr(i:i+1) = 0;
    Chiller_Pwr(i:i+1) = 0;
  end

  % Heater Communication
  if strcmp(Load_Priority,'Heater') == 0
    Heater_Pwr(i+1) = Heater_Pwr(i);
  end
  [Heater_Temp(i),Heater_Pwr(i+1)] =
Heater_Comm(COM,Heater_Pwr(i+1),FanSetting);

  % Chiller Communication
  if strcmp(Load_Priority,'Chiller') == 0
    Chiller_Pwr(i+1) = Chiller_Pwr(i);
  end
  if i > 9 && strcmp(Load_Priority,'Chiller') == 1
    if max(Chiller_Pwr(i-8:i+1)) > 0 && min(Chiller_Pwr(i-9:i)) == 0
      Heater_Comm(COM,0,'OFF');
      Heater_Pwr(i+1) = 0;
      Chiller_Pwr(i+1) = 22;
    end
  end
  Chiller_Comm(Chiller_Pwr(i+1));

  % Re-Print Header to Screen
  if mod(i,20) == 0;
    fprintf('Run \tTime  \tHeater\tStorage\tChiller\tStorage\tLoad\tPac
\tSolar\tWind\tBattery\tSOC \tO1\tO2\tO3 \n');
    fprintf('#  \tHH:MM:SS \t%%  \t%%  \t%%  \t%%  \tW  \tW  \tW  \tW
\tW  \t%% \n');
  end

  % Print Data to Screen
  fprintf('%1.0f  \t',i)
  fprintf(datestr(now,'HH:MM:SS\t'))
  fprintf('%3.0f \t%3.1f \t%3.0f \t%3.1f \t%4.0f \t%4.0f \t%4.0f
\t%4.0f \t%4.0f \t%2.1f \t%1.0f \t%1.0f \t%1.0f\n',...

Heater_Pwr(i),Heater_Storage(i),Chiller_Pwr(i),Ice_Storage(i),Pac_Load(
i),Pac(i),Pac_PV(i),Pac_WT(i),BatPower(i),BatSoc(i),Override_1(i),Overr
ide_2(i),Override_3(i));

  % Log Data
  fprintf(Control_Data,'%6.8f \t %3.0f \t %3.0f \t %4.0f \t %3.1f \t
%4.0f \t %4.0f \t %3.1f \t %3.1f \t %3.1f \t %4.0f \t %4.0f \t %4.0f \t
%4.0f \t %4.0f \t %4.1f \t %4.1f \t %4.0f \t %3.1f \t %4.0f \t %1.0f
\t%1.0f \t %1.0f \t\t %6.8f \t %2.1f \t %3.1f \t %3.1f \t %4.0f \t
%3.1f \t %3.1f \t %4.0f \t%3.1f \t %3.1f \t %4.0f \t %3.1f \t %4.0f \t
%4.0f \t\t %6.8f \t %3.1f \t %3.1f \t %4.0f \t %3.1f \t %3.1f \t %4.0f
\t %3.1f \t %3.1f \t %4.0f \t %3.1f \t %3.1f \t %4.0f \t %4.0f \t\t
%6.8f \t %3.1f \t %3.3f \t %2.2f \t %3.1f \t %3.1f \t %3.3f \t %4.0f \t
%4.0f \t %3.1f \t %3.3f \t %2.2f \t %3.1f \t %3.1f \t %3.3f \t %4.0f \t
%4.0f \t %4.0f \r\n',...

t(i),Heater_Pwr(i),Heater_Pwr_Max(i),Heater_Temp(i),Heater_Storage(i),C
hiller_Pwr(i),Chiller_Pwr_Max(i),Chiller_T_In(i),Chiller_T_Out(i),Ice_S
torage(i),Load(i),Pac_Load(i),Pac(i),Pac_PV(i),Pac_WT(i),BatVtg(i),TotB
atCur(i),BatPower(i),BatSoc(i),BatChrgOp(i),Override_1(i),Override_2(i)
,Override_3(i),t(i),InvFrq(i),InvVtg(i),InvCur(i),InvPwrAt(i),InvVtgSlv
1(i),InvCurSlv1(i),InvPwrAtSlv1(i),InvVtgSlv2(i),InvCurSlv2(i),InvPwrAt
Slv2(i),TotInvCur(i),TotInvPwrAt(i),TotInvPwrRt(i),t(i),AMsVol(i),AMsAm
p(i),AMsWatt(i),BMsVol(i),BMsAmp(i),BMsWatt(i),GridMsVphsA(i),GridMsAph
sA(i),GridMsWphsA(i),GridMsVphsB(i),GridMsAphsB(i),GridMsWphsB(i),Pac_P
V(i),t(i),Vpv_WT1(i),Ipv_WT1(i),Fac_WT1(i),Vac_L1_WT1(i),Vac_L2_WT1(i),
Iac_WT1(i),Pac_WT1(i),Mode_WT1(i),Vpv_WT2(i),Ipv_WT2(i),Fac_WT2(i),Vac_
L1_WT2(i),Vac_L2_WT2(i),Iac_WT2(i),Pac_WT2(i),Mode_WT2(i),Pac_WT(i));

  % Low Power Shutdown/Standby (2/2)
  if hour >= Start_Time && hour < Stop_Time;
    if Override_2(i) == 1
      if Override_2(i-1) == 0 || mod(i,5) == 0
        if Pac_ave(i) < Low_Power_Shutdown
```

```
          fprintf('\n SHUTDOWN: Average Power Generation %3.0f \n\n',
Pac_ave(i))
        elseif Pac_ave(i) < Low_Power_Standby && Pac_ave(i) >
Low_Power_Shutdown
          fprintf('\n STANDBY: Average Power Generation %3.0f \n\n',
Pac_ave(i))
        end
      end
    end
  end

  % Print "Control Startup/Shutdown" at Start and Stop Time
  if hour == Start_Time && minute == 0
    fprintf('\n CONTROL STARTUP \n\n')
  elseif hour == Stop_Time && minute == 0
    fprintf('\n CONTROL SHUTDOWN \n\n')
  end

  % Overnight Shutdown (2/2) - Pause Script
  if hour < Start_Time || hour >= Stop_Time
    Heater_Comm(COM,0,'OFF');
    Chiller_Comm(0);
%     pause(60*5); % If pause > 5 min, the webbox page will logout
  end

  % Restart Control Loop when Date Changes
  if i > 1
    if floor(t(i)) ~= floor(t(i-1))
      % Create New Data File
      fclose(Control_Data);
      Control_Data = fopen(['Control_Data_',date,'.txt'],'w');
      fprintf(Control_Data,'Time \t Heater \t Max \t Temp \t Storage \t
Chiller \t Max \t T_In \t T_Out \t Storage \t Load \t Load \t Pac \t
Solar \t Wind \t BatVtg \t BatCur \t BatPower \t SOC \t Mode \tO1 \t O2
\t O3 \t\t Time \t Fac \t Vac \t Iac \t Pwr \t Vac \t Iac \t Pwr \t Vac
\t Iac \t Pwr \t TotInvCur \t TotInvPwrAt \t TotInvPwrRT \t\t Time \t
Panels_A \t\t\t Panels_B \t\t\t Grid_A \t\t\t Grid_B \t\t\t Total \t\t
Time \t WT1_DC \t\t WT1_AC \t\t\t\t\t Mode \t WT2_DC \t\t WT2_AC
\t\t\t\t\t Mode \t Total \r\n');
      fprintf(Control_Data,'HHMMSS \t %% \t %% \t F \t %% \t %% \t %%
\t F \t F \t %% \t W \t W \t W \t W \t W \t V \t A \t W \t %% \t - \t -
\t - \t - \t\t HHMMSS \t Hz \t V \t A \t W \t V \t A \t W \t v \t A \t
W \t A \t W \t W \t\t HHMMSS \t V \t A \t W \t V \t A \t W \t V \t A \t
W \t V \t A \t W \t W \t\t HHMMSS \t V \t A \t Hz \t V \t V \t A \t W
\t - \t V \t A \t Hz \t V \t V \t A \t W \t - \t W \r\n');
      % Reset Variables
      Heater_Pwr     = zeros(1,10000);
      Heater_Pwr_Max = zeros(1,10000);
      Heater_Temp    = zeros(1,10000);
      Heater_Storage = zeros(1,10000);

      Chiller_Pwr     = zeros(1,10000);
      Chiller_Pwr_Max = zeros(1,10000);
      Chiller_T_In    = zeros(1,10000);
      Chiller_T_Out   = zeros(1,10000);
```

```
Ice_Storage   = zeros(1,10000);

Pac_Load    = zeros(1,10000);
TotInvPwrAt   = zeros(1,10000);
InvPwrAt    = zeros(1,10000);
InvPwrAtSlv1  = zeros(1,10000);
InvPwrAtSlv2  = zeros(1,10000);
InvVtg     = zeros(1,10000);
InvVtgSlv1   = zeros(1,10000);
InvVtgSlv2   = zeros(1,10000);
TotInvCur    = zeros(1,10000);
InvCur     = zeros(1,10000);
InvCurSlv1   = zeros(1,10000);
InvCurSlv2   = zeros(1,10000);
InvFrq     = zeros(1,10000);
TotInvPwrRt   = zeros(1,10000);
TotBatCur    = zeros(1,10000);
BatSoc     = zeros(1,10000);
BatVtg     = zeros(1,10000);
BatChrgOp    = zeros(1,10000);
Mode_WT1    = zeros(1,10000);
Ipv_WT1     = zeros(1,10000);
Vpv_WT1     = zeros(1,10000);
Pac_WT1     = zeros(1,10000);
Vac_L1_WT1   = zeros(1,10000);
Vac_L2_WT1   = zeros(1,10000);
Iac_WT1     = zeros(1,10000);
Fac_WT1     = zeros(1,10000);
Mode_WT2    = zeros(1,10000);
Ipv_WT2     = zeros(1,10000);
Vpv_WT2     = zeros(1,10000);
Pac_WT2     = zeros(1,10000);
Vac_L1_WT2   = zeros(1,10000);
Vac_L2_WT2   = zeros(1,10000);
Iac_WT2     = zeros(1,10000);
Fac_WT2     = zeros(1,10000);
AMsAmp     = zeros(1,10000);
AMsVol     = zeros(1,10000);
AMsWatt     = zeros(1,10000);
Pac_PV     = zeros(1,10000);
BMsAmp     = zeros(1,10000);
BMsVol     = zeros(1,10000);
BMsWatt     = zeros(1,10000);
GridMsWphsA   = zeros(1,10000);
GridMsWphsB   = zeros(1,10000);
GridMsVphsA   = zeros(1,10000);
GridMsVphsB   = zeros(1,10000);
GridMsAphsA   = zeros(1,10000);
GridMsAphsB   = zeros(1,10000);
Pac_WT     = zeros(1,10000);
Pac       = zeros(1,10000);
BatPower    = zeros(1,10000);
Load      = zeros(1,10000);
Pac_ave     = zeros(1,10000);
BatPower_ave  = zeros(1,10000);
```

```
    t         = zeros(1,10000);
    tplot       = zeros(1,10000);
    % Reset Overrides
    Override_1    = zeros(1,10000);
    Override_2    = zeros(1,10000);
    Override_3    = zeros(1,10000);
    % Turn ON MPP Override
    Override_1(1)   = 1;
    % Reset counter
    i = 1;
  else
    i = i + 1;
  end
else
  i = i + 1;
end

% Plot Control Data
if mod(i,15) == 0
  window = 180*6;
  if i > window
    x = t(i-window:i-1);
    y1 = Pac(i-window:i-1);
    y2 = Pac_Load(i-window:i-1);
    y3 = BatPower(i-window:i-1);
    y4 = Heater_Pwr(i-window:i-1);
    y5 = Chiller_Pwr(i-window:i-1);
    y6 = Heater_Storage(i-window:i-1);
    y7 = Ice_Storage(i-window:i-1);
    y8 = BatSoc(i-window:i-1);
  else
    x = t(1:i-1);
    y1 = Pac(1:i-1);
    y2 = Pac_Load(1:i-1);
    y3 = BatPower(1:i-1);
    y4 = Heater_Pwr(1:i-1);
    y5 = Chiller_Pwr(1:i-1);
    y6 = Heater_Storage(1:i-1);
    y7 = Ice_Storage(1:i-1);
    y8 = BatSoc(1:i-1);
  end
  figure(1)
  clf
  subplot(3,1,1)
  hold on
  plot(x,y1,'b')
  plot(x,y2,'r')
  plot(x,y3,'g')
  datetick('x','HH:MM')
  grid on
  grid minor
  xlabel('Time [HH:MM]')
  ylabel('Power [W]')
  legend('Pac','Load','Battery')
  subplot(3,1,2)
  hold on
  plot(x,y4,'r')
  plot(x,y5,'b')
  datetick('x','HH:MM')
  grid on
  grid minor
  xlabel('Time [HH:MM]')
  ylabel('Power [%]')
  legend('Heater','Chiller')
  subplot(3,1,3)
  hold on
  plot(x,y6,'r')
  plot(x,y7,'b')
  plot(x,y8,'g')
  datetick('x','HH:MM')
  grid on
  grid minor
  xlabel('Time [HH:MM]')
  ylabel('Storage [%]')
  ylim([0 100])
  legend('Heater','Chiller','Battery')
  pause(5)
end

% Pause Loop
loop_time = toc;
if loop_time < 20
  pause(20 - loop_time)
end

% End Control Loop
if loop_time > 40
  fprintf('\n Loop Timeout Error - Terminating Control Loop \n\n')
  try
    Heater_Comm(COM,0,'OFF');
    fprintf('\n Heater shut off \n\n')
  catch ME
    fprintf('\n')
    warning('Heater cannot be shut off')
    fprintf('\n')
    warning(ME.identifier,ME.message)
    fprintf('\n')
  end
  try
    Chiller_Comm(0);
    fprintf('\n Chiller shut off \n\n')
  catch ME
    fprintf('\n')
    warning('Chiller cannot be shut off')
    fprintf('\n')
    warning(ME.identifier,ME.message)
    fprintf('\n')
  end
  Run = 0;
end
```

```
    % Catch Matlab Errors - Shut Off Loads
    catch ME_loop
      fprintf('\n MATLAB Error - Terminating Control Loop \n\n ')
      try
        Heater_Comm(COM,0,'OFF');
        fprintf('\n Heater shut off \n\n')
      catch ME
        fprintf('\n')
        warning('Heater cannot be shut off')
        fprintf('\n')
        warning(ME.identifier,ME.message)
        fprintf('\n')
      end
      try
        Chiller_Comm(0);
        fprintf('\n Chiller shut off \n\n')
      catch ME
        fprintf('\n')
        warning('Chiller cannot be shut off')
        fprintf('\n')
        warning(ME.identifier,ME.message)
        fprintf('\n')
      end
      Run = 0;
    end

end

% Shut Off Loads
Heater_Comm(COM,0,'OFF');
Chiller_Comm(0);
% Close Data File
fclose(Control_Data);
% Close Modbus Port
fclose(WebboxTCP);
% Print to Screen
fprintf('\n CONTROL LOOP TERMINATED \n\n')
% Display Error
rethrow(ME_loop)
```

B. MICROGRID SAFETY SHUTDOWN

```
% Microgrid Controller Shutoff
clear;clc;

% Preallocate Memory
t         = zeros(1,10000);
file_size   = zeros(1,10000);

Run = 1;
i = 1;
while Run == 1
```

```
% Time
t(i) = now;
time = fix(clock);
hour = time(1,4);
minute = time(1,5);
fprintf(datestr(now,'HH:MM:SS\t'))

if hour ~= 0 || minute ~= 0
  % Try to open control data file
  try
    fid = fopen(['Control_Data_', date, '.txt']);
    Control_Data = fscanf(fid,'%c');
    fclose(fid);
    file_size(i) = length(Control_Data);
  catch ME
    warning('Data file does not exists')
    fprintf('\n')
    warning(ME.identifier,ME.message)
    fprintf('\n')
    try
      Heater_Comm(4,0,'OFF');
      fprintf('\n Large heater shut off \n\n')
    catch ME
      fprintf('\n')
      warning('Large heater cannot be shut off')
      fprintf('\n')
      warning(ME.identifier,ME.message)
      fprintf('\n')
    end
    try
      Heater_Comm(3,0,'OFF');
      fprintf('\n Small heater shut off \n\n')
    catch ME
      fprintf('\n')
      warning('Small heater cannot be shut off')
      fprintf('\n')
      warning(ME.identifier,ME.message)
      fprintf('\n')
    end
    try
      Chiller_Comm(0);
      fprintf('\n Chiller shut off \n\n')
    catch ME
      fprintf('\n')
      warning('Chiller cannot be shut off')
      fprintf('\n')
      warning(ME.identifier,ME.message)
      fprintf('\n')
    end
    Run = 0;
  end
  % Check if control data file is updating
  if i > 1 && Run == 1
    if floor(t(i)) == floor(t(i-1)) && file_size(i) == file_size(i-1)
      fprintf('Data file stopped updating \n')
    try
      Heater_Comm(4,0,'OFF');
      fprintf('\n Large heater shut off \n\n')
    catch ME
      fprintf('\n')
      warning('Large heater cannot be shut off')
      fprintf('\n')
      warning(ME.identifier,ME.message)
      fprintf('\n')
    end
    try
      Heater_Comm(3,0,'OFF');
      fprintf('\n Small heater shut off \n\n')
    catch ME
      fprintf('\n')
      warning('Small heater cannot be shut off')
      fprintf('\n')
      warning(ME.identifier,ME.message)
      fprintf('\n')
    end
    try
      Chiller_Comm(0);
      fprintf('\n Chiller shut off \n\n')
    catch ME
      fprintf('\n')
      warning('Chiller cannot be shut off')
      fprintf('\n')
      warning(ME.identifier,ME.message)
      fprintf('\n')
    end
    Run = 0;
    else
      fprintf('No Error \n')
    end
  else
    fprintf(' \n')
  end
end

% Restart Loop when Date Changes
if i > 1
  if floor(t(i)) ~= floor(t(i-1))
    i       = 1;
    t       = zeros(1,10000);
    file_size   = zeros(1,10000);
  else
    i = i + 1;
  end
else
  i = i + 1;
end

% Allow Control Data file to update (~20 sec)
pause(60)
```

```
  if Run == 0
    fprintf('\n Microgrid Safety Shutoff Terminated \n\n')
  end

  if mod(i,20) == 0
    clc
  end

end
```

C. MODBUS FUNCTION

```
function [Register_2] = Modbus(UnitID,Address,ModbusTCP)

% Function generates modbus message to poll given register

% Address - Address of the register you are trying to poll

% UnitID - UnitID of the inverter

% Quantity - Number of registers you want data from (2-125), values for
% specific variables are stored in 2 registers, the first register
% specifies whether the value is positive or negative, the second
register
% has the actual value

% ModbusTCP - TCP/IP object, establishing the connection
% Input prevents the function from re-establishing the connecting
% each time, which may causes Webbox to crash

  % SMA Sunny Webbox TCP/IP Object
  % ipaddress = '192.168.0.168';
  % port = 502;
  % ModbusTCP = tcpip(ipaddress, port);
  % set(ModbusTCP,'InputBufferSize',512);
  % ModbusTCP.ByteOrder = 'bigEndian';
  % fopen(ModbusTCP);

%% Default Inputs

Quantity = 2;

if nargin < 3
  ipaddress = '192.168.0.168';      % IP Address of the SUNNY WebBox
  port = 502;                % The port should be 502 for Modbus (cannot
be changed in the controller)
  ModbusTCP = tcpip(ipaddress, port);   % Create the tcpip object
  set(ModbusTCP,'InputBufferSize',512);  % assign the buffer
  ModbusTCP.ByteOrder = 'bigEndian';   % specify the order in which
bytes are transmitted
end

%% Try Opening Channel
```

```
  try
    if ~strcmp(ModbusTCP.Status,'open')
      fopen(ModbusTCP);
    end
  catch fault
    if ~strcmp(ModbusTCP.Status,'open')
      disp(fault);
      disp(['Error: Can''t establish TCP/IP connection with:
',ipaddress,':',num2str(port)] );
    end
  end

%% Create Data Request Message

  transID= int16(0);    % 16b transaction identifier
  ProtID = int16(0);    % 16b Protocol ID (0 for Modbus)
  Lenghf = int16(6);    % 16b Remaining bytes (24)
  UnitID = int16(256*UnitID); % Unit ID (1)
  FunCod = int16(03);    % Function code: read multiple input registers
(04)
  UnitIDFunCod = int16(UnitID + FunCod); % Concatenation of UnitID &
FunctionCode in one int16 word
  Add = int16(Address);   % 16b Address of the register
  Val = int16(Quantity); % 16b Data (read # registers)

  message = [transID; ProtID; Lenghf; UnitIDFunCod; Add; Val];

 %% Send Message and Process Response

  % Write message
  fwrite(ModbusTCP, message,'int16');

  % Check if message is received correctly
  while ~ModbusTCP.BytesAvailable
    % Wait for the response to be in the buffer
  end
  % Read received bytes
  result = fread(ModbusTCP,ModbusTCP.BytesAvailable);

  if result(8) < 128
    position1 = uint32(result(10)*256+result(11));
    position2 = uint32(result(12)*256+result(13));
%     position3 = uint32(result(14)*256+result(15));
%     position4 = uint32(result(16)*256+result(17));
%     position5 = uint32(result(18)*256+result(19));
  else
    position1 = 0;
    position2 = 0;
  end

  % Convert UINT32 to Double
  if double(position1) == 0
    Register_1 = double(position1);
    Register_2 = double(position2);
  else
    Register_1 = -1;
    Register_2 = -((double(position1) - double(position2)) + 1);
  end

end
```

D. BACNET FUNCTION

```
function [BACnetStatus,BACnetResults] =
BACnet(functionName,DeviceID,Object,Instance,Property,Value)

% Function executes BACnet communications

% functionName - BACnet function you wish to perform

% DeviceID   - ID of the device

% Object     - Analog Input = 0
%         - Analog Output = 1
%         - Analog Value = 2
%         - Binary Input = 3
%         - Binary Output = 4

% Instance   - Instance of object type on device

% Property   - Present Value   = 85
%         - Object Name    = 77
%         - Max Present Value = 65
%         - Min Present Value = 69
%         - Setpoint     = 108

% Value     - Value you want to write

%% Default Inputs

 if nargin < 2
   DeviceID = 0;
 end
 if nargin < 3
   Object = 0;
 end
 if nargin < 4
   Instance = 0;
 end
 if nargin < 5
   Property = 0;
 end
 if nargin < 6
   Value = 0;
 end
```

```
%% Change Inputs to Strings

 DeviceID  = num2str(DeviceID);
 Object   = num2str(Object);
 Instance  = num2str(Instance);
 Property  = num2str(Property);
 Value    = num2str(Value);

%% Establish Directory

 BACnetDir = ['C:\Users\Garth\Desktop\Hawxhurst\BACnet\bacnet-tools-
0.7.1' filesep];

%% Generate Address Cache

 if strcmp(functionName,'AddressCache')
   [BACnetStatus,BACnetResults] = dos([BACnetDir 'bacwi -1 >
address_cache']);
 end

%% List Devices

 if strcmp(functionName,'ListDevices')
   [BACnetStatus,BACnetResults] = dos([BACnetDir 'bacwi -1']);
 end

%% Read All Properties and Values

 if strcmp(functionName,'ListProperties')
   [BACnetStatus,BACnetResults] = dos([BACnetDir 'bacepics -v '
(DeviceID)]);
 end

%% Read

 if strcmp(functionName,'Read')
   [BACnetStatus,BACnetResults] = dos([BACnetDir 'bacrp ' (DeviceID) '
' (Object) ' ' (Instance) ' ' (Property)]);
   BACnetResults = str2double(BACnetResults);
 end

%% Write

 if strcmp(functionName,'Write')
   [BACnetStatus,BACnetResults] = dos([BACnetDir 'bacwp ' (DeviceID) '
' (Object) ' ' (Instance) ' ' (Property) ' 16 -1 4 ' (Value)]);
 end

end
```

E. HEATER COMMUNICATIONS FUNCTION

```
function [AvgTemp,ElementOnPercent] =
Heater_Comm(Port,ElementOnPercent,FanSetting)

% Function controls element on percentage and fan setting of heaters
% Outputs heater core temp and received element on percentage (should
be
% identical to input)

% ElementOnPercentage  - 0-100 in increments of 1

% FanSetting      - 'OFF','MEDIUM','HIGH','ON'

% Port         - USB Serial Port
%             - Small Heater = COM3(3)
%             - Large Heater = COM4(4)

%% Default Inputs

  if nargin < 2
    ElementOnPercent = 0;
  end
  if nargin < 3
    FanSetting = 'OFF';
  end

%% Change Inputs to Strings

  Port         = num2str(Port);
  ElementOnPercent  = num2str(ElementOnPercent);

%% Establish Port

  s = serial(['COM' (Port)]);
  fopen(s);
  set(s,'BaudRate',115200);
  set(s,'Terminator','CR/LF');

%% Read Temperature

  fprintf(s,'LO AvgTemp');
  AvgTemp = fscanf(s);
  AvgTemp = str2double(AvgTemp(7:10));

%% Set Element On Percent

  fprintf(s,['LO ElementOnPercent ',ElementOnPercent]);
  ElementOnPercent = fscanf(s);
  ElementOnPercent = str2double(ElementOnPercent(7:9));

%% Set Fan Setting
```

```
   if strcmp('OFF',FanSetting) == 1
    fprintf(s,'CB LV 0');
  elseif strcmp('MEDIUM',FanSetting) == 1
    fprintf(s,'CB LV 16');
  elseif strcmp('HIGH',FanSetting) == 1 || strcmp('ON',FanSetting) == 1
    fprintf(s,'CB LV 32');
  else
    fprintf(s,'CB LV 0');
  end

%% Close Port

  fclose(s);

end
```

F. CHILLER COMMUNICATIONS FUNCTION

```
function [ ] = Chiller_Comm(CompressorSpeedPercent)

% Function generates an analog output signal using NI cDAQ

% CompressorSpeedPercent  - 0-100 in increments of 1
%                  - turns off below ~20

%% Change Input to Voltage Signal

  volts_out = CompressorSpeedPercent/10;

%% Detect the data acquisition instrument

  devices = daq.getDevices;

%% Create Session

  s = daq.createSession('ni');

%% Add Analog Output Channel

  s.addAnalogOutputChannel('cDAQ9184-19114D1Mod2',0,'Voltage');
  s.addAnalogOutputChannel('cDAQ9184-19114D1Mod2',1,'Voltage');

%% Change Scan Rate (scans/second)

  s.Rate = 8000;

%% Generate Single Scan

  outputSingleValue = volts_out;
  s.outputSingleScan([outputSingleValue outputSingleValue]);

end
```

APPENDIX B. SMA MICROGRID COMMUNICATIONS SETUP

A. SUNNY WEBBOX WEBPAGE

- Connect the computer to the Sunny WebBox using an Ethernet cable
- Change the IP address of the computer to a fixed IP

IP Address:	**192.168.0.100**
Subnet Mask:	**255.255.255.0**

- Open an Internet browser and search IP address **192.168.0.168**, this should bring you to the Sunny WebBox login page

Username:	*****
Password:	*****

- From here, you can navigate using the plant dropdown menu

B. MODBUS

- Enable Modbus communications on the Sunny WebBox webpage

    WebBox > Network > Use Modbus > ‘yes’

    Modbus Port > ‘502’

- Check that all inverters are listed. If not, enter the number of devices that should be connected (4) and ‘start detection’

    Plant > Modbus

    Plant > Detection > Devices to be detected > ‘4’ > ‘Start detection’

- Assigned each device a Unit ID from 3–245. Do not use the same number twice

    Plant > Modbus > Unit ID > ‘3-245’

- Open MATLAB. Use the Modbus function to poll data from the inverters

    Modbus TCP – TCP/IP object of Modbus gateway (WebBox default)

    Unit ID – Unit ID of the desired inverter

    Register Address – Address of the register you want to poll

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. TRANE CHILLER COMMUNICATIONS SETUP

A. TRANE WEBPAGE

- Connect the computer to the Tracer SC controller using an Ethernet cable. Use either Ethernet network connection
- Open an Internet browser and enter **192.168.0.125**. This should take you to the Trane login page

Username:	**Tracer**
Password:	**TracerSC**

- From here, you can navigate the Trane chiller plant

B. NI CDAQ

- Turn the chiller breaker on
- Connect the NI cDAQ-9184 chassis to the computer
  - Load the NI 9201 (analog input) and NI 9262 (analog output) cards into the chassis
  - Connect the chassis to power
  - Connect the chassis to the computer using an Ethernet cable
- Open NI Max on the computer

  Devices and Interfaces > Network Devices > NI cDAQ-9184
- Check status indicates "Connected – Running" for the chassis and both cards
  - To test communications, select 'Self-Test'
- Connect the wires that run to the chiller control panel. The black wire (ground) screws into the second port (ground) on the NI 9262 card (analog out), the white wire connects to the top port (analog output) of the NI 9262 card. There are multiple analog output and ground ports.

- Open MATLAB. Type ‘daq.getDevices’
    - MATLAB should list the two cards, NI 9201 and NI 9263
    - If there is an asterisk next to the devices saying “Device currently not support” there is an issuer with the connection, go to NI Max and attempt to re-establish communications using “Self-Test”
- Troubleshooting
    - Run ‘Self-Test’ to re-establish connection. If self-test produces an error message and the status indicates “Disconnected,” the chassis may have been “Reserved” by another program.
    - Disconnect chassis > Restart computer (to un-reserve) > Reconnect chassis
- Additional Resources
    - MATLAB – Acquire Data Using NI Devices:

      http://www.mathworks.com/help/daq/examples/acquire-data-using-ni-devices.html?refresh=true
    - Network cDAQ Troubleshooting Resources (Step 7):

      http://digital.ni.com/public.nsf/allkb/E67B4E4749F378FF862577270059BD4B
    - cDAQ Firmware Update Instructions:

      http://www.ni.com/product-documentation/12778/en/#toc3
    - cDAQ Firmware 1.7.0 Download:

      http://www.ni.com/download/ni-cdaq-9184-firmware-1.7/5687/en/

C. BACNET

- Connect the computer to the Tracer SC controller using an Ethernet cable. Use Ethernet network connection 1 (this connection supports BACnet, connection 2 does not)
- Use the BACnet function on MATLAB to execute BACnet executables
  - ‘AddressCache’ – Discovers all BACnet devices on the network and stores addresses in a text file that can be used by the other function
  - ‘ListAllDevices’ – Discovers and lists all BACnet devices that are on the network
  - ‘Read’ – Specify the Device ID of the BACnet device and the Object, Instance, and Property of the variable you wish to read
  - ‘Write’ - Specify the Device ID of the BACnet device and the Object, Instance, Property, as well as the Value you wish to write to that variable
- Note: BACnet write commands are currently disabled by the Tracer SC controller

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. STEFFES HEATER COMMUNIATION SETUP

- Connect the heater to the computer using a USB cable
    - Control Panel > Devices Manager > Ports > COM3 (small heater) or COM4 (large heater)
    - If the USB cable does not appear as a COM Port, but an unknown USB, then the driver is not installed. Follow the FTDI Chip Driver installation guide to install the USB cable driver
- Power on the heater
    - For the small heater, plug the power cable into the power strip
    - For the large heater, turn on the breakers for the heater control panel and for the heater elements (4 breakers in total)
- Both heaters require ~60Hz AC power for the startup sequence to be completed successfully and will display an error message until this happens. The easiest way to do this is to turn on the chiller until the frequency drops to 60Hz. Once the heaters have turned on, the frequency doesn't matter.
- Open MATLAB. Using the correct Heater Comm function, you can control the Element on Percent and the Fan Setting
    - Element on Percent is a value from 0–100
    - Fan Setting are the string values ON (highest speed), MEDIUM, HIGH, or OFF (default)
- Additional Resources:
    - Cable Driver: http://www.ftdichip.com/Drivers/VCP.htm
    - Setup Executable:

      http://www.ftdichip.com/Drivers/CDM/CDM%20v2.12.00%20WHQL%20Certified.exe

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] D. Smalley. (2012, Oct. 18). Energize: ONR supports new energy partnership [Online]. Available: http://www.onr.navy.mil/Media-Center/Press-Releases/2012/Energy-Systems-Technology-ESTEP-ONR.aspx

[2] Environmental and Energy Study Institute. “DOD’s energy efficiency and renewable energy initiatives,” Environmental and Energy Study Institute, Washington, DC, 2011.

[3] Energy Department. (2014, Nov. 20). Top 9 things you didn’t know about America’s power grid [Online]. Available: http://energy.gov/articles/top-9-things-you-didnt-know-about-americas-power-grid

[4] Litos Strategic Communication, “The smart grid: An introduction,” U.S Dept. Energy, Washington, DC, (n.d.).

[5] S. Kaplan, “Electrical power transmission: Background and policy issues,” Congressional Research Service, Washington, DC, Rep. R40511, Apr. 2009.

[6] R. H. Lasseter, “MicroGrids,” in *IEEE Power Engineering Winter Meeting*, 2002, vol. 1, pp. 4285–4290.

[7] A. G. Pollman and A. J. Gannon, “Multi-physics energy approach & demonstration facility,” in *Power and Energy Conversion*, 2015, pp. 1–10.

[8] M. Schwartz and K. Blakeley, “Department of Defense energy initiatives: Background and issues for Congress,” Congressional Research Service, Washington, DC, Rep. R42558, Dec. 2012.

[9] HDIAC Staff. (2015, Aug. 12). Military’s shift toward renewable energy [Online]. Available: http://science.dodlive.mil/2015/08/12/militarys-shift-toward-renewable-energy/

[10] Assistant Secretary of Defense for Operational Energy Plans and Programs, “Operational energy strategy: Implementation plan,” Department of Defense, Washington, DC, Mar. 2012.

[11] R. George. (2015, Feb. 5). Defense Department energy use falls to lowest level since at least 1975 [Online]. Available: http://www.eia.gov/todayinenergy/detail.cfm?id=19871

[12] Marine Corps Expeditionary Energy Office. (n.d.). United States Marine Corps. [Online]. Available: http://www.hqmc.marines.mil/e2o/E2CExFOB.aspx

[13] J. Stamp, “Smart grid R & D program peer review meeting microgrid design toolset (MDT),” Sandia National Labs, Livermore, CA, 2014.

[14] Microgrids - smart grid solutions. (n.d.). Siemens. [Online]. Available: http://w3.usa.siemens.com/smartgrid/us/en/microgrid/pages/microgrids.aspx

[15] Office of Energy Efficiency and Renewable Energy. (n.d.). U.S. Dept. of Energy. [Online]. Available: http://energy.gov/eere/office-energy-efficiency-renewable-energy

[16] L. Olsen, “Initial investigation of a novel thermal storage concept as part of a renewable energy system,” M.S. thesis, Dept. Mech. Eng., Naval Postgraduate School, Monterey, CA, 2013.

[17] C. Borgnakke and R. Sonntag, *Fundamentals of Thermodynamics*, 7th ed. John Wiley & Sons, 2009.

[18] T. Hinke, “Hot thermal storage in a variable power, renewable energy system,” M.S. thesis, Dept. Mech. Eng., Naval Postgraduate School, Monterey, CA, 2014.

[19] D. Linden, *Handbook of Batteries*, 3rd ed. McGraw-Hill, 2002.

[20] Building Energy Data Book. (2012). U.S. Dept. of Energy. [Online]. Available: http://buildingsdatabook.eren.doe.gov/

[21] IceBank energy storage. (n.d.). CALMAC. [Online]. Available: http://www.calmac.com/icebank-energy-storage-model-c

[22] R. Boonyobhas, “Control strategy: Wind energy powered variable speed chiller with thermal ice storage,” M.S. thesis, Dept. Mech. Eng., Naval Postgraduate School, Monterey, CA, 2014.

[23] *SUNNY WEBBOX Modbus Interface*, SMA, Germany, 2014.

[24] *Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*, RS/ANSI/TIA/EIA Standard 232, 1969.

[25] Virtual COM port drivers. (n.d.). FTDI Chip. [Online]. Available: http://www.ftdichip.com/Drivers/VCP.htm

[26] BACnet. (n.d.). ASHRAE. [Online]. Available: http://www.bacnet.org/

[27] *BACnet and Modbus RTU for Trane Chillers with Tracer AdaptiView Control*, Trane, Dublin, Republic of Ireland, 2011, pp. 1–76.

[28] Tracer SC. (n.d.). Trane. [Online]. Available: http://www.trane.com/commercial/north-america/us/en/controls/building-Management/tracer-sc.html

[29] General controllers. (n.d.). Trane. [Online]. Available: http://www.trane.com/commercial/north-america/us/en/controls/HVAC-equipment-controls/general-controllers.html

[30] *AURORA Wind Box Interface Installation and Operator's Manual*, Power-One, San Jose, CA, 2006.

[31] *SUNNY BOY 5000-US / 6000-US / 7000-US / 8000-US*, SMA, Germany, (n.d.).

[32] Fleet numerical meteorology and oceanography center. (n.d.). U.S. Navy. [Online]. Available: http://www.public.navy.mil/fltfor/cnmoc/Pages/fnmoc_home.aspx

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California